The Dissertation Committee for Arvind Narayanan
certifies that this is the approved version of the following dissertation:

Data Privacy: the Non-interactive Setting

Committee:

Vitaly Shmatikov, Supervisor

Mike Dahlin

Anna Gal

Joan Feigenbaum

Ilya Mironov

Data Privacy: the Non-interactive Setting

by

Arvind Narayanan, B.Tech.; M.Tech.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2009

Acknowledgments

I'm grateful to my advisor Vitaly Shmatikov for making my life as a Ph.D student more pleasant than I thought possible. My best memories of my years here were the times I stayed up at his place until 4am arguing about global warming over glasses of wine. He was been a great guide and collaborator, letting me work on whatever research problem I wanted, putting aside everything else when tough deadlines needed to be met, and holding my hand over the less appealing aspects of research.

I would like to thank the members of my thesis committee for the feedback and advice they provided. Special thanks to Ilya Mironov, who was also my mentor at Microsoft Research; I have greatly enjoyed discussing my research with him.

I have been very lucky in having numerous research collaborators all over the country. Not only did this let me do virtually all my traveling at someone else's (mainly Vitaly's) expense, my abilities as a researcher have improved greatly by being exposed to other people's ideas and sub-fields of research. I also want to thank everyone who let me sleep on their couch during my travels. They are too many to list.

My parents, it would appear, decided when I was around 3 years old

that I would one day acquire a Ph.D. I have vivid and happy memories of soaking up arithmetic and language all day long from them at that age; in retrospect, my entire academic trajectory was determined 24 years ago! To them, to my sister Nivi — whom I could count on finding online whenever I was lonely, as my nocturnal habits neatly compensated for the difference in timezone — and the rest of my family who have been supportive, I am hugely indebted.

I enjoyed having my friend Justin Brickell as my office-mate very much. Not only did we bounce research ideas off each other, I would probably have been kicked out of the program long ago if it were not for Justin constantly reminding me to turn in some piece of paperwork or the other!

I am deeply grateful to Shweta Agrawal, to whom I owe more than words can describe.

I would like to thank Dmitry Sumin of Passware for providing the password material for the experiments reported in Chapter 4.

Cynthia Dwork introduced me to the problem of anonymity in social networks, for which I am very grateful. Kamalika Chaudhuri deserves special thanks for collaborating on an unpublished work on social-network anonymity; some of the broader themes carried over to the work described in this thesis. I have had many interesting discussions with Frank McSherry, Shuchi Chawla, Jason Davis, and many others.

This material is based upon work supported by the NSF grants IIS-0534198, CNS-0716158, and CNS-0746888 and the ARO grant W911NF-06-1-0316.

Arvind Narayanan

The University of Texas at Austin

May 2009

Data Privacy: the Non-interactive Setting

Publication No. _____

Arvind Narayanan, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Vitaly Shmatikov

The Internet has enabled the collection, aggregation and analysis of personal data on a massive scale. It has also enabled the sharing of collected data in various ways: wholesale outsourcing of data warehousing, partnering with advertisers for targeted advertising, data publishing for exploratory research, *etc.* This has led to complex privacy questions related to the leakage of sensitive user data and mass harvesting of information by unscrupulous parties. These questions have information-theoretic, sociological and legal aspects and are often poorly understood.

There are two fundamental paradigms for how the data is released: in the interactive setting, the data collector holds the data while third parties

interact with the data collector to compute some function on the database. In the non-interactive setting, the database is somehow "sanitized" and then published. In this thesis, we conduct a thorough theoretical and empirical investigation of privacy issues involved in non-interactive data release.

Both settings have been well analyzed in the academic literature, but simplicity of the non-interactive paradigm has resulted in its being used almost exclusively in actual data releases. We analyze several common applications including electronic directories, collaborative filtering and recommender systems, and social networks. Our investigation has two main foci. First, we present frameworks for privacy and anonymity in these different settings within which one might define exactly when a privacy breach has occurred. Second, we use these frameworks to experimentally analyze actual large datasets and quantify privacy issues.

The picture that has emerged from this research is a bleak one for non-interactivity. While a surprising level of privacy control is possible in a limited number of applications, the general sense is that protecting privacy in the non-interactive setting is not as easy as intuitively assumed in the absence of rigorous privacy definitions. While some applications can be salvaged either by moving to an interactive setting or by other means, in others a rethinking of the tradeoffs between utility and privacy that are currently taken for granted appears to be necessary.

Contents

Chapter 1

Introduction

The ease of large-scale data collection. Before the growth of the Internet, collection of data from individuals on a national or global scale was feasible only for governments and very large corporations. The infrastructure required for collecting and aggregating data was either in the form of a door-to-door survey as in a census, or a pervasive physical presence, such as a large supermarket chain collecting data on people's shopping habits. Clearly, the Internet has changed the equation dramatically: one might grasp the magnitude of the change by considering that the 2010 U.S. Census, which does not take advantage of information technology, is estimated to cost 14 billion USD, whereas a purely electronic survey of that scale, although hardly as rigorous, would cost orders of magnitude less.

Data sharing. The other emerging trend is sharing of collected data. There are several ways in which data collected about people is shared:

- Internet-scale data warehousing for tasks such as Customer Relationship Management is a specialized capability and is often outsourced, occasion-

ally to offshore providers. In addition, lower costs and higher reliability are cited as reasons for outsourcing.

- Highly targeted marketing and advertising are an important part of the business model of many Web 2.0 businesses. In fact, customer data allowing targeted advertising is considered the primary monetizable asset of companies such as Facebook. It is often necessary to share detailed customer data with advertisers for this purpose. Several companies such as Phorm, NebuAd and Front Porch are developing advertising systems based on behavioral targeting, which involve building detailed profiles of user behavior [258].

- Companies also make data available for exploratory research purposes. Such research is usually done in-house in corporations with dedicated research departments, but with the increasing ease of data collection by smaller companies, academia is often recruited for this purpose. Recent data releases by AOL [129] and Netflix [126] come to mind.

- Allowing users to search for and access information about other individuals is frequently an essential part of the functionality for which the data was collected. Social-network services are the most visible example of such functionality.

- More complex sharing situations arise when there is more than one party who owns a part of a dataset and joint analyses or computations need to be performed on the data. A joint medical study between two hospitals might involve sensitive patient data that needs to be aggregated to enable analysis. A network-security response center might aggregate audit log

data from dozens of institutions.

Finally, a truly distributed computation involves thousands or millions of users participating. Such systems are increasingly coming into fruition in the form of cloud computing, raising serious and largely unexplored privacy questions [130].

The factors described above have led to an explosion in the amount of personal data collected. This has given rise to complex privacy questions related to leakage of sensitive data on individuals and the mass harvesting of personal information by unscrupulous parties. Before we can discuss the privacy questions, however, it helps to have an understanding of the nature of the data being collected and the data-release processes.

1.1 Understanding data

1.1.1 Types of data

While all databases can be viewed simply in terms of rows and columns, it is useful to make further distinctions based on the semantics of the data.

Micro-data vs. aggregate data. Micro-data refers to data about individuals while aggregate data is information about a group of users or about the database as a whole. The terms originate from census terminology. Data when collected is always micro-data, whereas when sharing a database, one might choose not to make micro-data available. It is increasingly common to share micro-data because of the increasing complexity of the data being collected and the algorithmic complexity of the analyses that need to be performed. When micro-data is shared, it is often in an anonymized form.

High-dimensional data. The rows or records in a database typically respresent individuals, and the columns represent attributes. By dimensionality we mean the number of columns or attributes in the database. The terminology comes from viewing a record as a point in a vector-space. We empirically define high-dimensional databases as those for which algorithms that are exponential in the dimension are infeasible. While these could theoretically include any type of data, databases in current practice that we consider high-dimensional are those containing *transaction profiles* of users, which are vectors consisting of their preferences, behavior, purchase or transactional history for a variety of items of the same type.

Graph-structured data. Graph-structured data is derived from real-world networks of individuals and consists of nodes and edges, possibly with attributes corresponding to each node and edge. Since edges and edge-attributes cannot be associated with any single individual, a different abstraction becomes necessary.

Sensitive attributes. The semantics of the attributes sometimes make it meaningful to distinguish between sensitive and non-sensitive attributes. For instance, name and gender may not be sensitive but attributes pertaining to medical history might be. A similar notion is that of access vs. data attributes, for instance name vs. phone number in the context of a telephone directory. The statistical-database literature uses the notion of quasi-identifiers [69], which are attributes that are not structurally unique but potentially empirically unique, either by themselves or in combination with other quasi-identifiers, for instance ZIP code together with birth date in the context of a census database.

1.1.2 Release process

Interactive vs. non-interactive sharing. We can identify four categories of data sharing in the decreasing order of interactivity of the process. Traditionally, release of sensitive data involved manual auditing of each data access request to ensure compliance with privacy policies. For a recent example of HIPAA compliance guidelines involving manual auditing, see [200]. As privacy-preserving data mining techniques have matured—for example, the SuLQ framework [35] enables sophisticated query control and perturbation of query outputs—it has become possible to automate this process. This type of data sharing has not yet seen widespread adoption, as we will explain presently.

Moving further down on the scale of interactivity, the data collector might implement a lightweight query interface to the data, without additional safeguards such as output perturbation. Essentially, the data can be retrieved by crawling such a system. This allows some query control, but when such controls are not implemented, it can be viewed as equivalent to the final scenario, which is simply publishing the data, perhaps after "sanitization."

The appeal of non-interactivity. Non-interactive data release has been pursued aggressively in recent years because it avoids the costs and delays of implementing manual privacy safeguards. Compared to automated but interactive data sharing, non-interactive data release avoids the need to set up a reliable, high-performance, low-latency infrastructure for performing computations on databases. The added overhead of the privacy requirement means that interactive data release is often infeasible given current technological constraints.

In addition, rationing resources such as queries, computation and mem-

ory when there are multiple third parties interested in data analysis might very well prove insoluble in an interactive setting. Data mining often has a competitive aspect, even if it may not always be explicit. Genetic association studies are a good example [124].

Further, most work on interactive data mining does not address the requirement of privacy for the *client*, which is also important given the competitiveness of data mining. Thus, choosing an interactive setting might discourage interested third parties from participating.

1.1.3 Utility of data

Finally, we classify datasets based on the purpose for which they are shared. There are three categories that we will encounter: directory, data mining and statistical utility. This is not meant to be an exhaustive classification.

We use the term directory database or electronic directory in this work to refer to databases where the utility comes from the information associated with individual users (*e.g.*, a college alumni directory).

In a statistical database, the utility comes from learning statistical relationships (*e.g.*, census data). Typical uses are computing marginals, joint distributions and cross-attribute correlations. By data mining we mean various computations like clustering, classifiers, and collaborative filtering. The distinction between the latter two categories is fuzzy. In both of these categories, however, the utility comes from aggregate information and not individual data.

1.2 Protecting privacy

The nature of the privacy requirement depends on the intended utility of the data. In databases with an aggregate utility, the privacy constraint is that information associated with individuals should not be revealed. Directory databases, on the other hand, exist to make individual data available. However, one would like mass harvesting to be infeasible.

We now describe five broad strategies for protecting privacy. Note that these are not mutually exclusive.

Access control. Some form of access control is necessary for most datasets. The most common form of coarse-grained access control is passwords. Password-based authentication is often sufficient protection for data when the sharing requirements are not very complex. More fine-grained access control based on roles and privileges is an active research area and is available in commercial relational database systems [220, 205].

Query control can be a powerful tool for privacy protection in the interactive setting. It involves query filtering, such as allowing only SUM, COUNT and other aggregate queries; query logging and monitoring; and finally query auditing [6, 150]. Automated filtering of queries to ensure that responses do not leak sensitive information is hard; for instance, even denials might violate privacy. Technical solutions include simulatable auditing [146], but in many situations, occasional review of audit logs combined with authentication and the threat of punitive measures such as the revocation of query privileges might be sufficient to enforce self-policing of queries.

Perturbation-based techniques have a long history and include tools such as generalization, suppression, cell swapping and addition of noise. (How-

ever, the term perturbation is used in the literature to refer only to the last of these.) References may be found in Chapter 3.

Secure multi-party computation is a set of general cryptographic techniques that allow a set of players to compute joint functions of their inputs while leaking no information other than their respective outputs [264, 114]. It has been adopted to the problem of privacy-preserving data mining with some success [164, 163].

Anonymity is increasingly being used as a tool for protecting privacy in databases with aggregate utility in the non-interactive setting [241]. The rationale is that users are "de-identified", *i.e.*, identifying information about users is removed, then privacy is protected as long as the adversary cannot re-link the identity of an individual with their record in the database.

There are two reasons why anonymity has been especially popular as a privacy protection technique. First, de-idenitfication can be implemented easily and cheaply, as opposed to alternatives such as secure multi-party computation. Secondly, anonymized data release is often motivated by privacy laws. Even though such laws might not explicitly require anonymity as a precondition of release of sensitive data, they are often interpreted as such. We discuss this extensively in Section 7.3.4.

The use of de-identification as a "catch-all" privacy-protection mechanism, especially in the non-interactive data release scenario, is very tempting because it avoids the need to reason about the types of computations that users should (or should not) be able to perform on the released data. However, this thinking is flawed: privacy can only be meaningfully defined as a property of specific computation — as differential privacy does [82] — and not as a property of the data itself.

We believe that the dual trends of increasing non-interactivity in data sharing and the increasing reliance on (syntactically defined) anonymity for privacy protection have serious implications for privacy in data sharing. For this reason, much of this thesis is devoted to the analysis of anonymity (in particular, the threat of re-identification) in published data.

1.3 Applications

In this section we describe common applications where privacy is a concern in sharing data. These are the applications that this thesis focuses on.

Electronic directory. Electronic directory databases have a precursor in the form of telephone directories, with a long history going back to 1878 [67]. Interestingly, *reverse telephone directories* have been compiled for decades, although their availability to the general public has sometimes raised legal questions because of privacy issues.

Electronic directories potentially allow finer-grained functionality and privacy control. The operator may choose to make the existence of individuals in the database hidden unless there is enough information to identify them; access might be permitted based on certain fields and not on others; and mass harvesting may or may not be allowed. In fact, it is usually preferable to make mass harvesting infeasible. In the context of email, this concern is well known as the *directory harvest attack*; however, it is an issue in any electronic-directory setting.

This flexibility leads to some interesting and complex privacy issues. For instance, Facebook recently faced a privacy gaffe where users who chose to make the values of some attributes (such as religion and sexual orientation)

hidden found their privacy violated when their profiles showed up in searches based on specific values for those attributes [230].

Collaborative filtering. The term collaborative filtering refers to algorithms for predicting future user behavior by analyzing transactional profiles of a large number of users in conjunction. It is used primarily in online recommendation systems. Collaborative filtering algorithms operate, in an abstract sense, on a matrix (database) that contains a score for each (user, item) pair.

An alternate output of collaborative filtering algorithms is item-similarity information, *i.e.*, numerical similarity scores between each pair of items. This is useful, for instance, in deciding what items to put next to each other in a supermarket, or related items to show on a web page pertaining to the item currently being viewed

At an intuitive level, users do not like their purchase history being revealed, and yet would like to be able to feed this history into some system that is capable of making useful predictions. A formal framework for defining this notion is necessary. Purchase histories are usually protected by the privacy policy, and furthermore, there are often strong legal protections in place such as the Video Privacy Protection Act [87].

Social networks. Unique privacy challenges arise with sharing social-network data because the data is non-relational. Therefore, it is perhaps unsurprising that there currently exists no comprehensive framework for analyzing privacy and anonymity in social networks.

Online social-network services, which are a new development, face even more challenges because the data has both directory and aggregate purposes: users and application developers need access to the individual information of other users. On the other hand, researchers and advertisers need access

to aggregate information. For instance, Facebook alone has faced storms of privacy-related criticism of the way it shares data with users [243], applications [94], and advertisers [262].

Telephone-call graphs arguably contain much more sensitive information. The AT&T call graph, for instance, contains 1.9 trillion edges going back decades.[1] Law enforcement regularly mines information from such graphs. More worryingly, anonymized versions of call graphs are often published or shared for research purposes [5, 154].

The release of social-network data in anonymized form is hardly limited to the above situations, however. To give just one example, sexual relations between students in a number of high schools (totalling about 90,000 participants) have been collected as part of the Add Health dataset; the data from each school (often involving a thousand students) can be viewed as a social network. Such graphs are often published with identifying information removed [28].

[1] An edge in this context represents a single call.

Chapter 2

Contributions of this thesis

The purpose of this thesis is to investigate and understand privacy in non-interactive data sharing. We believe that a combination of theoretical and empirical methods is called for. On the one hand, there is currently a lack of rigorous privacy definitions and frameworks for analyzing privacy. Often, when doing a data release, there is no way of asserting that privacy is protected.

On the other hand, purely theoretical methods might produce algorithms that do not scale very well, or lead to definitions that are impossible to achieve in practice. Therefore, we will try to evaluate our definitions and algorithms as much as possible on real-world datasets.

Much of our work takes the form of studying fundamental problems with existing ways of sanitizing data. This consists of two parts: first, we provide a rigorous framework for defining privacy for a variety of applications; second, we provide an adversarial analysis of large datasets in each of these categories. There are striking similarities in these projects even though the data we are working with is very diverse: the Netflix prize dataset (Section 6) is a user-item similarity matrix. The social-network datasets (Section 7) are

graph-structured. The Amazon recommendations dataset (Section 8) consists mainly of an item-item similarity matrix. In every case, privacy vulnerabilities exist because of the fact that the adversary possesses *background knowledge* (sometimes referred to as *auxiliary information*), by which we mean knowledge about the individuals represented in the dataset obtained outside of the data release process under consideration.

In each of these analyses, we use a "scoring function" that associates a score with each user; the score represents the likelihood that a piece of background knowledge matches that user. The principles that go into building the heuristics for each scoring function are very similar. Each is tolerant to a large amount of *noise*, which refers to inconsistencies between the auxiliary information and the released dataset, whether accidental or deliberately introduced. Thus, we present a highly flexible, generic and robust algorithm for achieving de-anonymization. This reveals a fundamental inadequacy of the non-interactive data release paradigm. Section 2.2 discusses these and other common themes in greater detail.

2.1 Overview

2.1.1 The (in)security of passwords

Human-memorable passwords are a fundamental tool in protecting data in directory databases.[1] To decrease vulnerability of passwords to brute-force dictionary attacks, many organizations enforce complicated password-creation rules and require that passwords include numerals and special characters. We

[1]Of course, passwords are a mainstay of computer security in general; thus our work described in this section has a much broader impact than directory databases.

demonstrate that as long as passwords remain human-memorable, they are vulnerable to "smart-dictionary" attacks even when the space of potential passwords is large.

Our first insight is that the distribution of letters in easy-to-remember passwords is likely to be similar to the distribution of letters in the users' native language. Using standard Markov modeling techniques from natural language processing, it becomes possible to dramatically reduce the size of the password space to be searched. Our second contribution is an algorithm for efficient enumeration of the remaining password space. This allows application of time-space tradeoff techniques, limiting memory accesses to a relatively small table of "partial dictionary" sizes and enabling a very fast dictionary attack.

We evaluated our method on a database of real-world user password hashes. Our algorithm successfully recovered 67.6% of the passwords using a 2×10^9 search space. This is a much higher percentage than Oechslin's "rainbow" attack [199], which is the fastest known general technique for searching large keyspaces. These results call into question viability of text-based passwords as a privacy protection mechanism.

The key accomplishment in this work was the *enumeration algorithm for "Markovian" and regular language spaces, enabling a time-space tradeoff.* Chapter 4 (originally published as [192]) describes this work in detail.

2.1.2 Obfuscating directories

We investigate whether it is possible to encrypt an electronic directory database and then give it away in such a form that users can still access it, but only in

a restricted way. In contrast to conventional privacy mechanisms that aim to prevent *any* access to individual records, we aim to restrict the set of queries that can be feasibly evaluated on the encrypted database.

We start with a simple form of database obfuscation which makes database records indistinguishable from lookup functions. The only feasible operation on an obfuscated record is to look up some attribute Y by supplying the value of another attribute X that appears in the same record (*i.e.*, someone who does not know X cannot feasibly retrieve Y). We then (i) generalize our construction to conjunctions of equality tests on any attributes of the database, and (ii) achieve a new property we call *group privacy*. This property ensures that it is easy to retrieve individual records or small subsets of records from the encrypted database by identifying them *precisely*, but "mass harvesting" queries matching a large number of records are computationally infeasible.

Our constructions are non-interactive. The database is transformed in such a way that all queries except those explicitly allowed by the privacy policy become computationally infeasible, *i.e.*, our solutions do not rely on any access-control software or hardware.

The key accomplishment in this work was the *constuction of a secret-sharing-based cryptographic scheme for query control in the offline setting.* Chapter 5 (originally published as [193]) describes this work in detail.

2.1.3 Anonymity of sparse datasets

We present a framework for reasoning about anonymity for highly multidimensional databases containing anonymized records of individuals. We present an

algorithm that enables an attacker who has only a small amount of background knowledge about an individual to identify this individual's record if it is present in the database. We use our framework to reason about the efficacy of this algorithm. Our de-anonymization methods are robust to perturbation of attribute values in the published records, tolerate errors and fuzziness in the attacker's background knowledge, and work even in the situations where only a subset of the original database has been published.

We apply our de-anonymization methodology to the Netflix Prize dataset, which contains anonymous movie ratings of 500,000 subscribers of Netflix, Inc. We demonstrate that an attacker who knows only a little bit about an individual subscriber can easily identify this subscriber's record if it is present in the released dataset, or, at the very least, identify a small set of records which include the subscriber's record. This knowledge need not be precise, *e.g.*, the dates may only be known to the attacker with a 14-day error, the ratings may be known only approximately, and some of the ratings may even be completely wrong. Using the Internet Movie Database (IMDb) as our source of background knowledge, we successfully identified Netflix records of *non-anonymous* IMDb users, uncovering information—such as their apparent political preferences—that could not be determined from their public IMDb ratings.

The key accomplishmentss in this work were the *development of a privacy framework for sparse datasets, a de-anonymization algorithm tolerant to large amounts of noise and a numerical confidence measure of the result of the algorithm.* Chapter 6 (originally published in [194]) describes this work in detail.

2.1.4 Privacy in social networks

We propose a comprehensive mathematical model that provides an abstraction for the type of data collected in a social network, the background knowledge available to the adversary, the privacy policy, the data sanitization and release/sharing process, and what it means for the adversary to compromise privacy. Our privacy definition is builds on the notion of node re-identification.

We then present an attack that achieves node re-identification and thus violates anonymity and privacy. We identify a paradigm for de-anonymization consisting of two phases: *seed identification* and a feedback-based process that we call *percolation*, and discuss how each can be achieved. We discuss the implications of our attack in a world of increasingly portable and aggregated social-network data. We show how the defenses in the literature fail to stop our attacks. We motivate the use of centrality measures for measuring the effectiveness of attacks.

The key accomplishments in this work were the *development of a privacy framework for social networks, and a "self-correcting" algorithm that produces a mapping between two different network graphs based on a small amount of overlapping edge structure.* Chapter 7 (originally published in [195]) describes this work in detail.

2.1.5 Privacy of online recommender systems

Unlike the setting in Chapter 6 where the data is released in anonymized form, online recommender systems do not release transactional profiles of individual users in any form. However, two types of data are often available: the item-similarity matrix, and background knowledge consisting of a small number

of preferences in the form of publicly available reviews written by customers. We show a way of exploiting these two types of information to cause a privacy breach which leaks the purchases of users that are *not* part of the public record.

The key accomplishments in this work were *the reverse-engineering of the amazon.com recommendation system and the development of an algorithm for statistical deduction of customer purchases based on observation of item similarity lists.* Chapter 8 describes this work in detail; further information can be found in [48].

2.2 Themes

In this section we describe several common themes related to de-anonymization that arose in the course of this work. Broadly, these can be divided into conceptual and methodological themes. The conceptual themes relate to similarities in the types of data that we deal with and the relationship between the data and the amount of auxiliary information that the adversary needs. The methological themes pertain to the issues of validating and measuring the performance of the adversarial algorithms that we develop.

First, all of the large-scale real-world datasets that we work with (in Chapters 6, 7 and 8) are high-dimensional. The number of columns ranges from tens of thousands to millions.[2]

In addition to the total number of columns being large, the average number of non-null or non-empty columns in each record is also large, typically in the tens or hundreds. What this means is that each user has participated

[2]In the context of a social network, the dimensionality equals the number of nodes when we represent the graph as an adjacency matrix.

in a large number of transactions, which may include purchasing or reviewing items, designating other users as friends, etc. The average number of movies rated by a customer in the Netflix dataset is 213 (Chapter 6). Facebook claims an average degree of 120 in their network as of May 2009 [91]; the networks that we work with in Chapter 7 are smaller and have average degrees of around 30. In the subset of Amazon reviewers that we discuss in Chapter 8, the average number of reviews per user is 126.

One consequence of this fact is that the notion of similarity between users (or more precisely, user records) breaks down. According to [34], under a "broad set of conditions," as the dimensionality increases, the distance to the nearest data point approaches the distance to the farthest data point. The failure of commonly used anonymization techniques when the dimensionality of the data is large is explored from a theoretical perspective in [10].

We term datasets with this property *sparse*. A recurring theme throughout this work is that in a sparse dataset, a very small number of attributes of a user are sufficient to uniquely identify his record in the dataset. Theoretical justification for this fact is provided in Theorems 5, 6 and 7 in Chapter 6—we find that a *logarithmic* number[3] of attributes to be sufficient under a variety of assumptions.

Numerically, between 2 and 8 attributes seem to suffice for movie rating data in Chapter 6; the seed-identification algorithm in Chapter 7 makes accurate inferences using no more than 10 individual pieces of information; in Chapter 8, we find that predictions of the form "user X bought item Y" can be made based on (roughly) 9 pieces of auxiliary information about past user purchases. In work not reported in this thesis, we observe the same

[3]Logarithmic in the number of records in the dataset.

principle in a dataset published in anonymized form by Lending Club [191]. Golle and Partridge find that just two pieces of geo-location information about an individual—home and work location, to the granularity of a U.S. Census block—are uniquely identifying on average [117].

As noted above, the effect of the size of the dataset on the amount of auxiliary information required is weak, *i.e.*, logarithmic. The *computational* requirements increase linearly (Chapter 6) or in some cases super-linearly (Chapter 7) as the size of the dataset increases. However, we do not believe that computational requirements of are a serious impediment to de-anonymization, because the size of the dataset is upper bounded, regardless of the application, by the total human population, which is currently less than 2^{33} and grows at a far slower rate than the adversary's computing power.

The final conceptual theme that we would like to discuss is the fact that the term "Personally Identifiable Information" (PII), commonly used in privacy law and breach disclosure law, has no particular technical meaning. Algorithms that can identify a user in an anonymized dataset are agnostic to the semantics of the data elements. While some data elements may be uniquely identifying on their own, any element can be identifying in combination with others. Our de-anonymization work in this thesis has repeatedly demonstrated this fact. In Section 7.3.4, we present a detailed discussion of laws that mention PII.

Turning to methological issues, the primary hurdle that any demonstration of a de-anonymization algorithm must overcome is the lack of "ground truth," *i.e.*, knowledge of the true identities of the users that have been re-identified. Ground truth is needed in order to verify that the algorithm has succeeded. Note that if ground truth is available as part of the auxiliary infor-

mation, re-identification is trivial, obviating the de-anonymization algorithm!

In Chapter 6, we develop a novel solution to this problem. We propose a measure called *eccentriticy*, which measures, given an anonymized dataset and auxiliary information about one of the records, how well the record that most closely matches the auxiliary information "stands out" from the rest of the records. We use this technique to find matches in the anonymized Netflix dataset of movie ratings that are as many as 28 standard deviations away from the second-best match, giving a high degree of confidence in the output of the de-anonymization algorithm.

We take an alternate approach in Chapter 7. We construct the ground-truth mapping between the two social networks that we investigate as part of the large-scale deanonymization exercise. We do this using information such as username, real name and location reported by users on the two networks. The de-anonymization algorithm itself entirely ignores the information used to construct the ground-truth mapping, and attempts to match nodes between two graphs where there is no identifying information attached to the nodes. While it is true that the existence of ground truth information means that our experiments did not cause actual de-anonymization (in the sense of recovering previously unknown identities), we present a detailed list of scenarios in Section 7.2 where our algorithm is applicable and can in fact cause an anonymity breach.

The use of the eccentricity measure has an important side-effect. It allows the de-anonymization algorithm to determine if a match is "spurious," *i.e.*, the auxiliary information corresponds to a record that is not actually in the database. This is a pressing concern whenever the released dataset is only a sample from a larger dataset, *i.e.*, it does not encompass all the individuals

from whom data was collected.

To recap, the eccentricity heuristic protects against two sources of error: one, a false match, whereby where the matching record is not the "true" record that corresponds to the auxiliary information, and two, a spurious match, where the "true" record is not even a part of the released dataset. Our numerical results clear demonstrate its effectiveness: from Figure 6.5, it can be seen that the adversary can simultaneously achieve a false positive rate and a false negative rate of under 10%, with only a slight increase in the amount of auxiliary information required.

This leads us to conclude that data trustees are often mistaken when making claims that anonymously published data is safe because some of the samples were withheld. An example of such a mistaken claim is this quote from the International HapMap Project [214]:

> The samples are anonymous with regard to individual identity. Samples cannot be connected to individuals, and no personal information is linked to any sample. As an additional safeguard, more samples were collected from each population than were used, so no one knows whether any particular person's DNA is included in the study.

Another major hurdle that any de-anonymization algorithm must overcome is *noise*. By noise we mean inconsistencies between the anonymized data and the auxiliary information available to the adversary. There are many sources of noise. Accidental inaccuracies are one obvious possibility—auxiliary information typically comes from users participating in multiple systems that collect the same data, and the data is often provided inaccurately or incom-

pletely.

Systemic differences are a more serious source of noise—two systems might collect broadly similar but subtly different data. For instance, in the case of movie ratings, one system might collect the date when the user watched the movie, and another might collect the date when they reviewed it. In the case of social networks, one expects the friendships of an individual to be similar across different networks, but not the same, due to the differing semantics of friendship.

The final category is noise that is deliberately injected into the anonymized dataset that is released, as part of the sanitization process. Such sanitization has been shown to be less effective than is often assumed, due to the inevitable destruction of the utility of the data [45]. Our experience mirrors this finding: in our analyses of real-world datasets, noise due to systemic differences and inaccurate reporting were by far the more prominent sources.

We have succeeded in developing algorithms that are significantly more robust to noise than previous de-anonymization efforts. For instance, Frankowski *et al.*, who also de-anonymize movie rating data, report that their algorithm is easily foiled by discrepancies between the auxiliary information and the anonymized record [102]. By contrast, our algorithm tolerates different types of noise as well as a higher level of noise (Section 6.4.1, Figure 6.10). In Chapter 7, we find that we can identify users across social networks even though the overlap in their relationships on the two networks is only 15%.

A major reason why our algorithms are robust to noise is that are built by combining many heuristics. It may not always be clear which heuristics will be effective before testing on real-world data; at the same time, there are many general principles that have guided us. A good example is the effect

of the rarity of an attribute: we have found that an attribute contributes to de-anonymization in proportion to the inverse logarithm of the frequency of its occurence (see, for instance, Section 6.4). The intuitive justification for this is that the entropy required to describe an attribute is inversely proportional to the logarithm of its frequency.

Chapter 3

Overview of related work

3.1 Password security

Research on password security predating our work used both Markov modeling and time-space tradeoffs. The "Extensible Multilingual Password Generator" software, written by Jon Callas in 1991, used precisely this technique to *generate* passwords for users. The password-guessing rules used in John the Ripper could be considered a form of finite-automaton-based enumeration. There is a long line of research on time-space tradeoff techniques for password cracking, starting from Hellman [134] and Rivest [74], continuing with [95, 155, 156] and culminating in the so-called rainbow crack of Oechslin [199, 228, 1] which our work builds on.

There are two main categories of defenses against the sort of attacks that we come up with. The first is to move away from alpha-numeric passwords, for instance graphical passwords (see [141] or [237] for a survey) and passwords based on mouse movement, facial recognition, and keystroke dynamics [186].

These systems face an uphill battle for adoption: first, because their usability is not well-established [58], and second, because they often fall prey to attacks [116, 72].

A much more promising alternative is "Password-based Authenticated Key Exchange" (PAKE) [32, 33, 169, 44, 31, 143] which essentially bootstraps a low-entropy passphrase into a cryptographic key. Recent work [111, 110] has eliminated limitations such as the need for multiple servers and the need to store the passphrase in plaintext on the server.

From the perspective of an attacker (passive or active) listening on the network, PAKE removes the ability to carry out offline attacks. While online attacks have the same complexity as offline attacks, they are far harder because the server is "in the loop," and therefore, can mount defenses such as rate-limiting the number of connections. This might enable the attacker to cause a denial of service, but this is presumably a much less serious attack than discovering passwords.

Section 4.1 discusses related work on password security in greater detail.

3.2 Database obfuscation

Our work on database obfuscation applies the cryptographic notion of obfuscation to the database privacy problem. The notion of obfuscation was first comprehensively studied by Barak *et al.* [27]. The obfuscation of "point functions" in the random oracle model, while perhaps a folklore construction, was first stated formally by Lynn, Prabhakaran and Sahai [170].

There are several lines of work that tackle variants of the problem. Perhaps the closest in spirit to our work is attribute-based encryption [223].

It achieves fine-grained access control by granting different access rights to different users. Recent work such as [118] achieves the ability to implement flexible policies where decryption keys are associated with access structures.

In the online setting, the harvesting problem can be solved by rate throttling. This also largely ameliorates the dictionary-attack vulnerability. However, it introduces the new problem of privacy for the client. Oblivious keyword search [103] is a way to solve the problem of privacy for the client.

Section 5.2 discusses related work on obfuscation in greater detail.

3.3 Privacy-preserving data mining

There is a very long line of research on statistical disclosure control and privacy-preserving data mining. It can be broadly divided into three categories in the increasing order of rigor: informally specified privacy, formally but non-adversarially specified privacy, and finally adversarially defined privacy. These categories roughly correspond to the statistics, database, and cryptography research communities.

Traditional work used techniques such as cell suppression, quantization, aggregation, perturbation, swapping, and sampling [79, 259, 96]. Typically, privacy is specified informally in these papers. Another strategy involves reconstructing statistical models of the data instead of sharing micro-data. Later work focused on techniques like k-anonymity and l-diversity [241, 172] and other data-oriented privacy definitions [11, 13, 12, 89].

Focusing on adversarially defined privacy, traditional cryptographic work on Secure Multi-party Computation [265, 114] has been adapted to solve some specific privacy-preserving data mining problems [92, 209]. In the interac-

tive setting, the SuLQ framework [35] provides strong privacy guarantees. There are various impossibility results, both in the interactive and in the non-interactive model [77, 81]. Finally, differential privacy has emerged as a powerful definition in both models. The definition has been adapted to a variety of settings and applications [179, 106, 82] and has matured from theory to practice in a very short period of time.

There is very little work that focuses on high-dimensional databases, as can be seen from the paucity of papers on privacy-preserving collaborative filtering. Even papers with "multi-dimensional" in the title involve algorithms that are exponential in the dimension [158].

k-anonymity also fails on high-dimensional data [10], since it fundamentally relies on locality of data. In fact, the expected distances to the nearest and farthest neighbors are almost the same for high-dimensional data under a variety of distributions [34].

Previous work on anonymity of movie ratings (which we consider in Chapter 6) reached similar, if weaker conclusions as we did: Frankowski *et al.* demonstrated in [102] that it is possible to correlate public mentions of movies in the MovieLens discussion forum with the MovieLens users' movie rating histories in the *internal* MovieLens dataset.

Section 6.2 discusses related work on anonymity of transactional data in greater detail.

3.4 Privacy and anonymity in social networks

Research on social networks is roughly a century old [104]. The study of social networks form a key part of epidemiology [19, 187], sociology [119, 248], and

economics [41, 25], among many other disciplines.

The formal study of privacy in social networks appears to be fairly new. Backstrom, Dwork and Kleinberg present "active attacks" (where the attacker is capable of modifying the network) and a weaker passive attack with the goal of recovering edge relations [23]. A few papers have proposed defenses against these and similar attacks: Hay *et al.* study perturbation as a tool to defend against specific re-identification attacks [131]. Liu and Terzi propose modifying the graph to anonymize degree sequences [167]. We note that both these defenses have fairly limited scope. In another vein, Jagatic *et al.* report an interesting experiment where the availability of friend relationships between users to the attacker dramatically increases the success of phishing [139].

Hanneman and Riddle provide a good introduction to centrality and power from a sociology perspective [128]. The work of Bonacich and Lloyd takes a more algorithmic approach [38]. Kempe, Kleinberg and Tardos consider the complexity of the problem of selecting the most influential nodes in a network [145].

Section 7.3 discusses related work on privacy and anonymity in social networks in greater detail.

3.5 Privacy of online recommender systems

To our knowledge, Ramakrishnan *et al.* were the first to discuss privacy risks in recommender systems. Their focus is on "straddlers", *i.e.*, customers who purchase a variety of rare items in different categories [216]. Their analyses pertain to the model where the recommendations made to the users—rather than item-similarity scores—leak sensitive information. This is a significant

limitation because the former type of data is much harder for an attacker to obtain.

There are two papers that tackle privacy issues in collaborative filtering using a secure-multi-party-computation-like protocol with each participant keeping control of their own data [54, 212].

McSherry and Mironov have recently developed a technique for providing provable privacy guarantees in collaborative filtering based on a relaxed definition of differential privacy [178]. While the accuracy of the output of the system is somewhat diminished, the authors evaluate their approach on the Netflix Prize dataset and find that an accuracy that is superior to the Cinematch baseline can be achieved while providing a reasonable level of privacy.

Linden *et al.* provide a description of amazon.com's recommendation system [165] (Chapter 8). Further information is found in a patent [166].

Chapter 4

Insecurity of Human-Memorable Passwords

4.1 Introduction

It is often said that humans are always the weakest link in the security chain. Social engineering is more frequently successful in penetrating a system than buffer overflow attacks. Similarly, cracking a password is rarely accomplished by breaking the cipher used and more often by exploiting the circumstance that it was generated by a human.

The problem of guessing human-generated passwords has been studied for a long time. Morris and Thompson [188] in their 1979 paper on UNIX password security describe brute-force and dictionary attacks that are all too familiar to the modern reader. The former is based on the observation that short strings are easier for humans to remember, and the latter on the fact that meaningful words are far more memorable than random character sequences.

The state of the art in dictionary attacks has not advanced much since then. Current password crackers such as John the Ripper [204] use essentially the same two techniques, plus a few rules to turn each dictionary word into a set of related words (such as suffixing a digit). The brute-force attack can be made somewhat more efficient by using a time-space tradeoff such as Oechslin's "rainbow" technique [199, 1], which employs precomputation to speed up the process of cracking individual passwords. A common defense is the use of password-creation rules, called *composition rules*, that require passwords to be drawn from certain regular languages, and require passwords to include digits and non-alphanumeric characters. The goal is to increase the size of the search space, making naive dictionary attacks infeasible.

We demonstrate that fast, devastating "smart-dictionary attacks" can be staged even on large password spaces that are not vulnerable to brute-force or straightforward dictionary attacks. Our insight is that passwords, even if long and sprinkled with extra characters, must still be human-memorable. Human limitations imply that the entropy of passwords is rather small: an NIST document [47] estimates that user-generated 8-character passwords have between 18 and 30 bits of randomness, even when English dictionary words are filtered out and composition rules are enforced. In a very general sense, our attacks work by rapidly enumerating candidate passwords that can be produced with a small amount of randomness. For instance, the string `asasasasasasasasasasasasasasasas`, at 32 characters, is far beyond the reach of brute-force attacks using current hardware. Intuitively, however, this string is not very random, and should be easily guessable.

Formally, this can be modeled by saying that the *Kolmogorov complexity* [161] of this string is low. The Kolmogorov complexity (a.k.a. K-

complexity) of a string is defined as the length of the shortest Turing Machine that outputs it and then halts. The sequence of instructions
`repeat print "as" 16 times` in any reasonable encoding scheme should have a complexity of around 30 bits, which is in keeping with our intuition that the resulting string is easily guessable.

An obvious approach would be to try to enumerate all strings which have a K-complexity smaller than some threshold, and use the result as the dictionary. There are two problems with this. First, K-complexity is uncomputable. Second, human perception of randomness in different from computational randomness. Subjective randomness has been studied by cognitive scientists Griffiths and Tenenbaum [121, 122], who argue that we need a different way to measure the perceived information content of a string. What is needed is a rigorous multidisciplinary study of human password generation processes, as well as the algorithms for very efficient enumeration of the resulting password spaces. The existence of such algorithms would be a strong indicator of vulnerability to fast dictionary attacks.

We take the first steps in both directions. We posit that phonetic similarity with words in the user's native language is a major contributor to memorability. We capture this property using "Markovian filters" based on Markov models, which is a standard technique in natural language processing [215]. Our other observation is that the way in which humans use non-alphabetic characters in passwords can be modeled by finite automata, which can be considered a generalization of the "rules" used by John the Ripper and other password cracking tools.

Secondly, we present an algorithm to *efficiently* enumerate all strings of a given length that satisfy a Markovian filter, and an algorithm to efficiently

enumerate all strings of a given length accepted by a deterministic finite automaton. More precisely, we give algorithms that, given an index i, generate the i^{th} string satisfying each of these properties (without searching through the entire space!). Next, we combine these two algorithms into a hybrid algorithm that produces the i^{th} string satisfying *both* the Markovian filter and the regular language. Finally, we show how to use this algorithm to implement a time-space tradeoff over this search space.

The best-known brute-force attack technique using the time-space tradeoff was proposed by Oechslin [199]. It uses a special data structure called the "rainbow table." Our hybrid attack has the same precomputation time, storage requirement and mean cryptanalysis time as the rainbow attack over a search space of the same size. Yet, compared to the vanilla rainbow attack over the entire fixed-length keyspace (*e.g.*, "all alphanumeric strings of length 8"), our attack has a significantly higher *coverage* since our search space is chosen wisely and consists only of "memorable" strings that have low subjective randomness.

Of course, coverage of a dictionary attack can only be measured by applying it to "real-world" user passwords. Results obtained on artifically generated passwords would not be very convincing, since there is no guarantee that the generation algorithm produces passwords that are similar to those users choose for themselves. We evaluated our techniques on a database of 150 real user passwords provided to us by Passware. Our hybrid attack achieved 67.6% coverage (*i.e.*, more than two thirds of the passwords were successfully recovered), using a search space of size 2×10^9. Detailed results are presented in Section 4.5. The filters we used can be found in Appendix A. By contrast,

Oechslin's attack achieves coverage of only 27.5%[1].

We believe that with a larger search space (around 3×10^{12}, which is what the `passcracking.com` implementation uses), and more comprehensive regular expressions, coverage of up to 90% can be achieved.

Defenses against dictionary attacks. Salting of password hashes defeats offline dictionary attacks based on precomputation, and thus foils our hybrid attack.

Using an inefficient cipher slows the attacker down by a constant factor, and this is in fact done in the UNIX `crypt()` implementation. This technique, however, can only yield a limited benefit because of the range of platforms that the client may be running. Javascript implementations in some browsers, for example, are extremely slow.

Feldmeier and Karn [93] surveyed methods to improve password security and concluded that the only technique offering a substantial long term improvement is for users to increase the entropy of the passwords they generate. As we show, achieving this is far harder than previously supposed.

By far the strongest line of defense against our attacks is Password-Authenticated Key Exchange (PAKE) [32, 169, 31, 44, 143, 113, 111, 109, 270, 108]. The goal of PAKE is to enable participants to "bootstrap" a shared password into a high-entropy cryptographic session key. Early PAKE protocols were found to have flaws [149, 207, 263], but recent ones offer provable security and have been standardized as RFC [136].

To understand how PAKE protects against our attacks, we must analyze

[1]We used the character set consisting of lowercase alphabets and numerals for Oechslin's attack. Simple tricks can improve the coverage, such as running more than one timespace tradeoff with different character sets, and using brute force up to some small length.

the points of vulnerability in a typical scenario where a user authenticates himself to a server over the network.

- The attacker may observe the authentication process by listening on the network, and use the transcript of the interaction to launch an *offline* attack.

- The attacker may try to discover the password in an *online* fashion, by repeatedly executing the client side of the authentication protocol using various password guesses as input.

- The attacker may break into the server where authentication material is stored; this may give him a "password verification string," using which he can launch an offline attack.

A good PAKE protocol has the property that the transcript of a successful authentication does not provide the attacker with a way to verify password guesses. Note that it is impossible to prevent an attacker who breaks into the server from obtaining password verification material, because access to the storage on the server allows the attacker to simulate both the client side and the server side of the authentication protocol in an offline fashion.

Thus, if PAKE is used, there are two ways in which our smart-dictionary attack may still be mounted: either via an online attack, or via a server break-in. Neither of these is as powerful an attack mode as an offline attack using the network transcript. There are a number of heuristics to protect against online password-guessing attacks, such as rate-limiting of protocol instantiations and blacklisting IP addresses with suspected malicious activity. These measures

come at a cost, because they may enable the attacker to cause a denial of service to legitimate users.

Needless to say, PAKE protects against attacks involving a time-space tradeoff—compared to salting, PAKE is a much more powerful way of randomizing the stored material on the server.

PAKE is not useful for authenticating a user on a local system, since there is no network activity to protect against. In conclusion, PAKE fails to mitigate against our attack in some situations (local login), prevents the time-space tradeoff component of our attack in others (network login), and makes even the smart-dictionary attack without precomputation much harder in yet other situations (network login involving no server compromise).

The above discussion assumes that PAKE can be easily implemented wherever applicable. This is far from the case, due to the cost of upgrading legacy systems—software needs to be upgraded, and more importantly, users must re-enter their passwords into the system. Re-entry of passwords is necessary because the server-side password verification material in PAKE protocols cannot be derived from the hashed passwords that are typically stored on the server.

In the absence of a generic defense, our attacks call into our question whether it is meaningful for humans to generate their own character-sequence passwords. The situation can only become worse with time because hardware power grows exponentially while human information-processing capacity stays constant [182]. Considering that there is a fundamental conflict between memorability and high subjective randomness, our work could have implications for the viability of passwords as an authentication mechanism in the long run.

Organization of this chapter. In Section 4.2, we explain filtering based on Markovian models and deterministic finite automata. In Section 4.3, we discuss the time-space tradeoff in general and Oechslin's rainbow attack in particular, and show that it works over *any* search space as long as there is an efficient algorithm to compute its i^{th} element. In Section 4.4, we present our search space enumeration algorithms for strings satisfying a Markovian filter, strings accepted by a DFA and strings that satisfy *both* conditions. Section 4.5 consists of our experimental results. A summary is found in Section 4.6.

4.2 Filtering

In the rest of this chapter, we will use the words *key* and *password*, *ciphertext* and *password hash*, and *keyspace* and *dictionary* interchangeably. The reason for this is that standard techniques for brute-force cryptanalysis, such as those described in Section 4.3, typically refer to keys and ciphertexts even when applied to passwords and their hash values.

4.2.1 Markovian filtering

An alphabetical password generated by a human, even if it is not a dictionary word, is unlikely to be uniformly distributed in the space of alphabet sequences. In fact, if asked to pick a sequence of characters at random, it is likely that an English-speaking user will generate a sequence in which each character is roughly equidistributed with the frequency of its occurrence in English text. Analysis of our password database reveals a significant number of alphabetical passwords which are neither dictionary words, nor random sequences (we used the `openwall.com` dictionary which contains about 4 million words [203]).

Markov models are commonly used in natural language processing, and are at the heart of speech recognition systems [215]. A Markov model defines a probability distribution over sequences of symbols. In other words, it allows sampling character sequences that have certain properties. In fact, Markov models have been used before in the context of passwords. Subsequent to this work, we learned of the "Extensible Multilingual Password Generator" software, which was written by Jon Callas in 1991 and used precisely this technique to *generate* passwords for users. (We are also told that a similar method for generating "random, yet pronounceable" passwords was used at the Los Alamos National Laboratory in the late 1980s.) It should come as no surprise, then, that Markov modeling is very effective at guessing passwords generated by users.

In a *zero-order* Markov model, each character is generated according to the underlying probability distribution and independently of the previously generated characters. In a *first-order* Markov model, each digram (ordered pair) of characters is assigned a probability and each character is generated by looking at the previous character. Mathematically, in the zero-order model,

$$P(\alpha) = \Pi_{x \in \alpha} \nu(x)$$

while in the first-order model,

$$P(x_1 x_2 \ldots x_n) = \nu(x_1) \Pi_{i=1}^{n-1} \nu(x_{i+1} | x_i)$$

where $P(.)$ is the Markovian probability distribution on character sequences, x_i are individual characters, and the ν function is the frequency of individual

letters and digrams in English text.

Of course, a dictionary is not a probability distribution, but a set. Therefore, to create a Markovian dictionary, we discretize the probabilities into two levels by applying a threshold θ. This defines the zero-order dictionary

$$\mathcal{D}_{\nu,\theta} = \{\alpha : \Pi_{x\in\alpha}\nu(x) \geq \theta\}$$

and the first-order dictionary

$$\mathcal{D}_{\nu,\theta} = \{x_1x_2\ldots x_n : \nu(x_1)\Pi_{i=1}^{n-1}\nu(x_{i+1}|x_i) \geq \theta\}$$

The zero-order model produces words that do not look very natural, but it can already drastically reduce the size of the plausible password space by eliminating the vast majority of character sequences from consideration. Consider 8-character sequences. If θ is chosen so that the dictionary size is $\frac{1}{7}$ of the keyspace (*i.e.*, 86% of sequences are ignored), then a sequence generated according to the model has the probability of 90% of belonging to the dictionary. In other words, 15% of the password space contains 90% of all *plausible* passwords. Other interesting points on the curve are: a dictionary containing $\frac{1}{11}$ of the keyspace has 80% coverage, and a dictionary with $\frac{1}{40}$ of the keyspace has 50% coverage, *i.e.*, only 2.5% of the keyspace needs to be considered to cover half of all possible passwords! The first-order model can do even better. The results are shown in Figure 4.1.

The zero-order model is not obviated by the first-order model. The former is better for certain commonly used password-generation strategies, such as acronyms consisting of first letters of each word in a sentence.

Needless to say, the distribution of letter frequencies used in keyspace

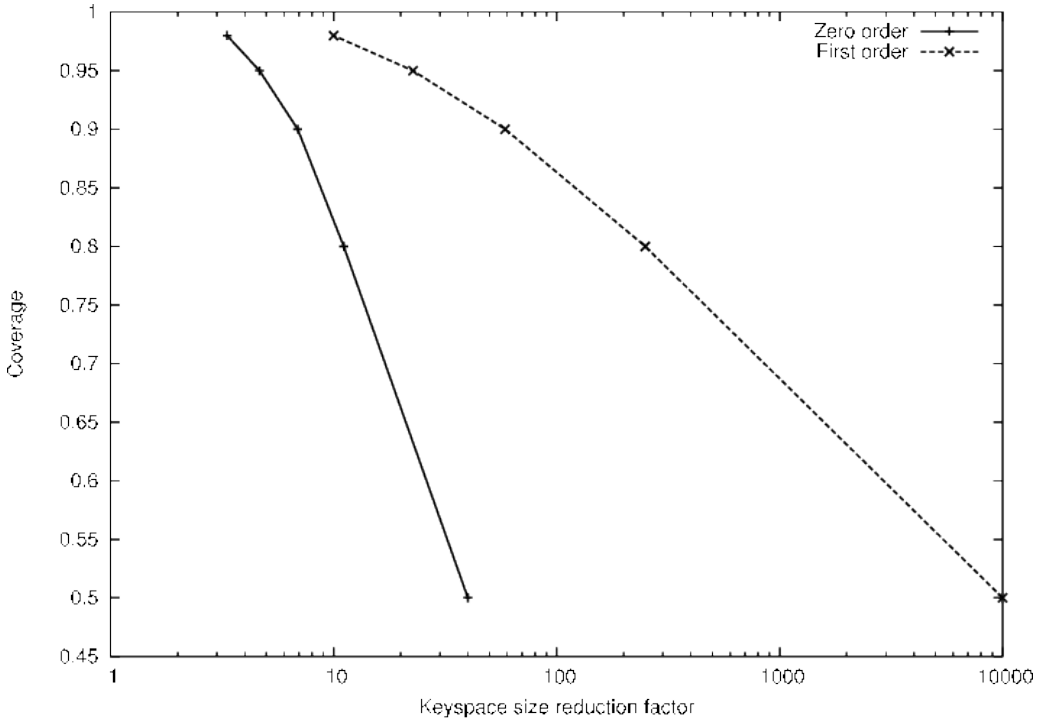

Figure 4.1: Coverage compression graph

compression via Markovian filtering is language-specific, and the distribution used in this work applies only to passwords chosen by English-speaking users (presumably, similar distributions can be found for other alphabet-based languages). There are two ways in which the technique can be generalized if the user's native language is not known. First, it is possible to combine the keyspaces for two or more languages (Section 4.4.6). Second, it may be possible to come up with a distribution that works reasonably well for multiple languages (*e.g.*, all Germanic or all Romance languages). We have not tried doing this and don't know how well it might work.

4.2.2 Filtering using a finite automaton

A search space consisting of only alphabetic sequences is unlikely to have a good coverage of the plausible password space. Humans often mix upper- and lowercase characters in their passwords, and system-enforced password-creation rules often require them to throw in some numerals and sometimes special characters. Yet, even with these additions, the distribution of resulting passwords is far from random. Below are examples of a few common patterns (this list is by no means definitive):

- In an alphanumeric password, all numerals are likely to be at the end.
- The first character of an alphabetic sequence is far more likely to be capitalized than the others.
- While alphabetic sequences consisting of mostly lowercase characters and a few uppercase characters are common, the converse is not true.

Deterministic finite automata are ideal for expressing such properties. First, we specify a set of common regular expressions ("all lowercase," "one uppercase followed by all lowercase," "uppercase characters followed by numerals," and so on). We define our dictionary to be the set of sequences matching the Markovian filter *and* also accepted by at least one of the finite automata corresponding to the regular expressions. Thus $\mathcal{D}_{\nu,\theta,\langle M_i\rangle} = \{\alpha : \Pi_{x\in\alpha}\nu(x) \geq \theta, \text{ and } \exists i : M_i \text{ accepts } \alpha\}$. Our complete alphabet consists of 26 lowercase and 26 uppercase characters, 10 numerals and 5 special characters (space, hyphen, underscore, period and comma). We have chosen these five somewhat arbitrarily because we felt they are the ones that occur most commonly in human-generated passwords. It is not possible to consider *all* special characters without knowing what the actual character set is for the application.

This gives us a 67-character alphabet. The associated keyspace of 8-letter sequences is 10^{15}, making brute-force search infeasible. While it is possible to write regular expressions for this 67-character symbol set, the resulting algorithms are not very efficient. Therefore, we give up a little bit of expressivity and group the character set into four categories (lowercase, uppercase, numerals, and special characters, which we will denote, respectively, as a, A, n and s), and consider the input alphabet of our automata to consist of just these four symbols. The regular expressions we have used are listed in Appendix A.

4.3 Time-space tradeoff

The most basic precomputation technique is to compute and store the hashes of all the passwords in the keyspace ordered by hash, so that cryptanalysis

is almost instantaneous. However, this requires storage equal to the keyspace size. Hellman [134] showed how to decrease storage requirements at the expense of attack time in the general context of cryptanalyzing a cipher. This tradeoff works as follows.

Given a fixed plaintext P, define a mapping f on the keyspace $\mathcal{K}$ as $f(k) = R(E_k(P))$ where E is the encryption function and R is a *reduction function* which maps ciphertexts to keys. By iterated applications of f, we create a "chain" of keys. The crucial observation is that by storing *only the first and last elements of a chain*, we can determine if the key corresponding to a given ciphertext belongs to that chain (and also find the key) in time $O(t)$ where t is the length of the chain.

Creating a chain works as follows. Given a starting point k_0, we compute $k_1 = f(k_0), k_2 = f(k_1), \ldots, k_t = f(k_{t-1})$. The keys k_0 and k_t are stored, while the rest are thrown away. When given a ciphertext C to cryptanalyze, we recover the key by computing $k = R(C)$ and $k_i = f^{i-1}(k)$ for $i = 1, 2, \ldots$. Observe that if the key k belongs to the chain, *i.e.*, $k = k_i$ for some i, then $f^{t-i}(k) = k_t$. Thus, after at most t applications of f, we can determine whether or not the chain contains k. Further $i - 1$ applications of f suffice to compute k_{i-1} from k_0. Since k_{i-1} satisfies the property that $C = E_{k_{i-1}}(P)$, it is the key we are looking for.

This does not work with certainty because different chains may merge. Therefore, we need to use multiple tables with a different reduction function for each table, with many chains in each table. The storage requirement as well as the cryptanalysis time for this algorithm are $O(|\mathcal{K}|^{2/3})$, where $\mathcal{K}$ is the keyspace.

Rivest [74, p.100] suggested an improvement which greatly speeds up

the practical performance of the algorithm by decreasing the number of memory accesses during cryptanalysis. This is done using "distinguished points," which are keys with some special property (such as the first 10 bits being zero). The advantage of requiring that the endpoints of a chain be distinguished points is that during cryptanalysis, a key must be looked up in memory only if it has the distinguished property. Further papers [95, 155, 43, 156, 233] presented optimizations and/or better analyses of the distinguished points algorithm.

A major improvement was made by Oechslin [199] by using "rainbow chains" instead of distinguished points. Rainbow chains use a different reduction function for each point in the chain. It was shown in [199] that rainbow chains achieve the same coverage as distinguished points with the same storage requirement but a significantly faster cryptanalysis time.

Experimental results are impressive. The online implementation of the rainbow attack [1] inverts MD5 hashes of passwords of length up to 8 over the character set `[a-z0-9]` (a keyspace size of 2.8×10^{12}). Its success probability is 0.996 with an amortized cryptanalysis time of under 10 minutes using precomputed tables of size about 48 GB.

4.3.1 Generic time-space tradeoff using index lookup property

We observe that in both Rivest's and Oechslin's algorithms, the reduction function can be expressed as a mapping from the ciphertext space to $\{0, 1, \ldots |\mathcal{K}| - 1\}$, composed with a mapping from $\{0, 1, \ldots |\mathcal{K}| - 1\}$ to $\mathcal{K}$. Neither the reduction function, nor any other part of the algorithms makes any assumptions

about the keyspace other than its size. In Rivest's attack, the property of being a distinguished point can be computed from the keyspace index rather than the key itself. In the case of the rainbow attack, the mapping from the ciphertext space to keyspace indices is parameterized by the choice of the rainbow table, but the mapping from the keyspace index to the key is the same as in Rivest's attack. Therefore, we can implement either attack over the compressed dictionary of plausible passwords described in Section 4.2.

This is not trivial, however. The compressed dictionary must have have the property that there exists an *efficient enumeration algorithm* which takes index i as input and outputs the i^{th} element of the dictionary. In the next section, we present such indexing algorithms for zero-order Markovian dictionaries, first-order Markovian dictionaries, deterministic finite automata (DFA), an arbitrary keyspace with some indexable superspace, and, finally, for hybrid Markovian/DFA dictionaries.

4.4 Indexing algorithms

4.4.1 Zero-order Markovian dictionary

The dictionary we use is a slightly modified version of the zero-order Markovian filter, in which we only consider fixed-length strings. This is because we want to use different thresholds for different lengths. The hybrid algorithm in Section 4.4.5 will demonstrate how multiple dictionaries can be combined into one. The modified dictionary is $\mathcal{D}_{\nu,\theta,\ell} = \{\alpha : |\alpha| = \ell \text{ and } \Pi_{x\in\alpha}\nu(x) \geq \theta\}$

The key to the algorithm in this section is discretization of the probability distribution. To do this, we first rewrite the filter in an additive rather

than multiplicative form: $\mathcal{D}_{\nu,\theta,\ell} = \{\alpha : |\alpha| = \ell \text{ and } \sum_{x\in\alpha} \mu(x) \geq \lambda\}$ where $\mu(x) = \log \nu(x)$ and $\lambda = \log \theta$.

Next, we discretize the values of μ to the nearest multiple of μ_0 for some appropriate μ_0. If we use a larger value for μ_0, we lower our memory requirement, but lose accuracy as well. In our experiments, we have chosen μ_0 such that there are about 1000 "levels" (see below).

Next, we define "partial dictionaries" $\mathcal{D}_{\nu,\theta,\ell,\theta',\ell'}$ as follows. Let α be any string such that $|\alpha| = \ell'$ and $\Pi_{x\in\alpha}\nu(x) = \theta'$. Then $\mathcal{D}_{\nu,\theta,\ell,\theta',\ell'} = \{\beta : \alpha\beta \in \mathcal{D}_{\nu,\theta,\ell}\}$.

Note that $\mathcal{D}_{\nu,\theta,\ell,\theta',\ell'}$ is well-defined because

$$\Pi_{x\in\alpha\beta}\nu(x) = \Pi_{x\in\alpha}\nu(x)\Pi_{x\in\beta}\nu(x) = \theta'\Pi_{x\in\beta}\nu(x)$$

Therefore, it doesn't matter which α we choose. Intuitively, for any string prefix, the partial dictionary contains the list of all possible character sequences which could be appended to this prefix so that the resulting full string satisfies the Markovian property.

We now present a recursive algorithm to compute the size of a partial dictionary

$$|\mathcal{D}_{\nu,threshold,\ total\_length,\ level,\ current\_length}|$$

Observe that this algorithm is executed only once and only during the precomputation stage (rather than once for each key), and, therefore, its efficiency does not affect the cryptanalysis time. Here `mu` refers to the discretized version of the μ function above.

```
partial_size1(current_length, level)
```

```
{
    if level >= threshold: return 0
    if total_length = current_length: return 1

    sum = 0
    for each char in alphabet
        sum = sum + partial_size(current_length+1,
            level+mu(char))
    return sum
}
```

Computation of $\mathcal{D}_{\nu,\theta,\ell,.,\ell'}$ depends on $\mathcal{D}_{\nu,\theta,\ell,.,\ell'+1}$. Thus the partial sizes are computed and stored in a 2-D array of size ℓ times the number of levels, the computation being done in the decreasing order of ℓ'.

We are not particularly concerned about the efficiency of this algorithm because it is executed only during precomputation. Note, however, that running time (for computing all partial sizes) is linear in the product of the total length, number of characters and the number of levels.

The following is another recursive algorithm which takes as input an index into the keyspace and returns the corresponding key (this algorithm is executed during the cryptanalysis stage):

```
get_key1(current_length, index, level)
{
    if total_length = current_length: return ""

    sum = 0
```

```
    for each char in alphabet
        new_level = level + mu(char)
        // looked up from precomputed array
        size = partial_size1[
            current_length+1][new_level]
        if sum + size > index
            // '|' refers to string concatenation
            return char | get_key1(
                current_length+1,
                index-sum, new_level)
        sum = sum + size

    // control cannot reach here
    print "index larger than keyspace size"; exit
}
```

The `get_key` algorithm uses `partial_size` to determine the first character (this results in a value being looked up in the precomputed table of partial sizes), and then recurs on the index recomputed relative to the first character and the threshold adjusted based on the frequency of the first character.

To index into the entire keyspace, we call `get_key1` with `current_length` = 0 and `level` = 0.

We note the similarity of the ideas used in this algorithm to the well-known Viterbi algorithm from speech processing [99].

4.4.2 First-order Markovian dictionary

As in the case of the zero-order model, we define length-restricted dictionaries and their partial versions. After reading a partial string, however, we now need to keep track of the last character because this time we are using digram frequencies.

```
partial_size2(current_length, prev_char, level)
{
    if level >= threshold: return 0
    if total_length = current_length: return 1

    sum = 0
    for each char in alphabet
        if current_length = 0
            new_level = mu(char)
        else
            new_level = level + mu(prev_char, char)
        sum = sum + partial_size2(current_length+1,
            char, new_level)
}

get_key2(current_length, index, prev_char, level)
{
    if total_length = current_length: return ""

    sum = 0
    for char in alphabet
```

```
        if current_length = 0
            new_level = mu(char)
        else
            new_level = level + mu(prev_char, char)
        size = partial_size2(current_length+1,
            char, new_level)
        if sum + size > index
            return char | get_key2(
                current_length+1,
                index-sum, char, new_level)
        sum = sum + size

    // control cannot reach here
    print "index larger than keyspace size"; exit
}
```

4.4.3 Deterministic finite automaton

This algorithm is similar to the algorithm for zero-order Markovian dictionaries, except that instead of levels and character frequencies we have states and state transitions. The `get_key3` algorithm is very similar to `get_key1` and is omitted.

```
partial_size3(current_length, state)
{
    if current_length = total_length
        if state is an accepting state: return 1
```

```
        else: return 0

    sum = 0
    for char in alphabet
        new_state = transition(char, state)
        if new_state is not NULL
            sum = sum + partial_size3(
                current_length+1, new_state)
    return sum
}
```

4.4.4 Any keyspace

We now describe an indexing algorithm for any keyspace $\mathcal{K}$, which works as long as there is an indexing algorithm for some superspace $\mathcal{K}' \supset \mathcal{K}$ and a testing procedure which, given $\alpha \in \mathcal{K}'$, decides whether $\alpha \in \mathcal{K}$. For instance, we can trivially index into (unfiltered) character sequences of a given length and test if a character sequence satisfies a Markovian filter, and, therefore, we can use this algorithm to index into Markovian dictionaries. The disadvantage is that precomputation involves enumerating $\mathcal{K}'$ via its indexing algorithm which might be prohibitively expensive if $\mathcal{K}$ is sparse in $\mathcal{K}'$. For instance, it is quite reasonable to consider 10-character password sequences with a first-order Markovian filter. This compresses the keyspace by a factor of 10^5, but to use the algorithm below to achieve this would involve iterating over a keyspace larger than 10^{14}. Furthermore, the indexing itself is not very efficient. On the other hand, it provides a good starting point because further keyspace-specific

optimizations may be possible.

Given a parameter t, the algorithm divides the space $\mathcal{K}'$ into bins of size t, precomputes the number of members of $\mathcal{K}$ in each bin and stores them. When it gets an index, it quickly figures out which bin it falls into, iterates over all keys in $\mathcal{K}'$ in that bin and tests each one for membership.

Let $|\mathcal{K}'| = mt$.

```
compute_bins(t)
{
    count=0
    for i = 0 to m-1
        for j = i*t to i*t+(t-1)
            if the j'th key of K' belongs to K
                count = count+1
        bin[i] = count
}
```

For each i, this computes the cumulative counts for the first i bins.

```
get_key(index)
{
    i = binary_search(bin[], index)
    // i.e., bin[i] < index <= bin[i+1]

    count=0
    for j = i*t to i*t+(t-1)
        key = the j'th key of K'
            if key belongs to K
```

```
            count++
        if count = index - bin[i]
            return key
}
```

This algorithm requires $O(|\mathcal{K}'|)$ precomputation time, $O(\frac{|\mathcal{K}'|}{t})$ storage and $O(t + \log \frac{|\mathcal{K}'|}{t})$ indexing time. Observe that this algorithm is very similar to the algorithm for finding the nth prime [40].

4.4.5 Hybrid Markovian/DFA dictionary

Let $\mathcal{A}$ be the set of characters, and consider the combined dictionary $\mathcal{D}_{\nu,\theta,\ell_1,M,\ell_2} = \{\alpha : |\alpha| = \ell, \sum_{x\in\alpha, x\in\mathcal{A}} = \ell', \Pi_{x\in\alpha, x\in\mathcal{A}}\nu(x) \geq \theta, \text{ and } M \text{ accepts } \alpha\}$.

As mentioned earlier, our finite automaton works over the symbol set $\{A, a, n, s\}$. All lowercase characters are represented by `a`, all uppercase characters by `A`, all numerals by n and all special characters by s in the input to the automaton.

```
get_key5(index)
{
    count1 = partial_size1(0, 0)
    count2 = partial_size2(0, initial_state)

    index1 = index/count2  // quotient is truncated
    index2 = index - index1 * count2

    key1 = get_key1(0, index1, 0)
    // wlog we assume that key1 consists of
```

```
    // lowercase characters
    key2 = get_key2(0, index2, initial_state)

    key = ""
    pos = 1
    for char in key2:
        if char is 'a'
            append key1[pos] to key
            pos = pos+1
        if char is 'A'
            append uppercase(key1[pos]) to key
            pos = pos+1
        if char is neither 'a' nor 'A'
            append char to key

    return key
}
```

Essentially, this algorithm looks at the positions in the output of `get_key2` where alphabet characters are expected, and substitutes the string returned by `get_key1`. Combining an automaton with a first-order Markovian filter works similarly.

4.4.6 Multiple keyspaces

Finally, we show how multiple disjoint keyspaces can be combined into a single space.

```
get_key6(K1, K2, ... Kn, index)
{
    sum = 0
    for i = 1 to n
        if sum + size(Ki) > index
            return get_key(Ki, index-sum)
        sum = sum + size(Ki)

    // this cannot be reached
    print "index larger than sum of keyspace sizes"
}
```

4.4.7 Possible optimizations

In this section, we describe some optimizations that we did not implement. The main criterion for the hybrid attack is that indexing should take less time than the hashing algorithm. So we want the hybrid algorithm to use about $50 - 100$ table lookups and the table must fit into the cache. The automaton algorithm is very fast because its input alphabet is very small; `get_key6` can be slow when there is a large number of keyspaces, but it can accelerated by precomputation of the cumulative sums and binary search for the appropriate keyspace. Our main concern, then, are the Markovian filters. We can speed up `get_key1` by reordering the characters so that the more frequent ones come first, reducing the average number of iterations to 6 (from 13, if the characters are ordered alphabetically). This reduces table lookups to less than 50 for 8-character strings. The same strategy works for `get_key2`, too.

4.5 Experiments

Our first experiment involves measuring the coverage of Oechslin's rainbow attack vs. our hybrid attack. We used a database containing 142 real user passwords kindly provided by Passware (operator of `http://LostPassword.com`), which we believe to be a representative source. We used the search space of 6-character alphanumeric sequences (lowercase characters only) for the rainbow attack, which gives a keyspace size of $36^6 = 2.17 \times 10^9$. To model common password patterns, we created a set of around 70 regular expressions, which are listed in Appendix A.

The following table compares the number of passwords recovered by the Rainbow attack vs. our attack.

Category	Count	Rainbow	Hybrid
Length at most 5	63	29	63
Length 6	21	10	17
Length 7	18	0	10
Length 8, A* or a*	9	0	6
Others	31	0	0
Total	142	39(27.5%)	96(67.6%)
only length ≥ 6	79	10(12.7%)	33(41.8%)

The effect of the probabilistic nature of the time-space tradeoff has been neglected, since the probability can be arbitrarily increased by increasing table size (and the dependence is the same for both attacks).

This experiment validates our basic hypothesis, but further experiments are needed. Passwords in our database may not have been representative of typical user passwords due to the way the database was compiled by Pass-

ware. We had access to the passwords when creating our regular expressions, although we did take the utmost care to write the expressions without using specific knowledge about the passwords in the database. A better experiment would involve a database containing only hashes of passwords, *e.g.*, the contents of an `/etc/passwd` file obtained from a system with a large, diverse user base.

4.6 Summary

There are a variety of attacks against passwords, of which dictionary-based attacks are only one subclass. The simplest to deploy are social engineering attacks such as impersonation, bribery, phishing and login spoofing. Other attacks that directly exploit human vulnerabilities include shoulder surfing and dumpster diving. Password-based authentication systems appear particularly susceptible to protocol weaknesses, which can be exploited by keystroke logging, "Google hacking," wiretapping and side-channel attacks based on timing and acoustic emanations. Among dictionary-based attacks, it is worth mentioning that the United States Secret Service recently reported [147] success with custom dictionaries built from victim-specific information gleaned from analyzing their hard drives, including their documents, email messages, web browser cache and contents of visited websites.

Defending against dictionary attacks on human-memorable passwords is a difficult task. Possible techniques include graphical passwords [141, 246, 72], reverse Turing tests [210, 236], and hardening passwords with biometric information [186]. These techniques, however, require substantial changes in the authentication infrastructure. An interesting question for future research

is whether human-memorable passwords drawn from non-textual spaces (*e.g.*, faces or geometric images) are vulnerable to attacks such as ours, based on filtering out unlikely candidates and very efficient enumeration of the remaining ones. Investigating this question will only be possible after one of the proposed methods is widely adopted and there is a significant body of such passwords chosen by real users.

Online dictionary attacks are generally considered easier to defend against than offline attacks. This work provides more evidence for this belief: the speedup of our smart-dictionary attack using a time-space tradeoff works only in the offline setting. To the extent possible, we recommend preventing the possibility of offline attacks, such as by implementing PAKE rather than naive hash-based password authentication.

Chapter 5

Obfuscated Databases and Group Privacy

5.1 Introduction

Conventional privacy mechanisms usually provide all-or-nothing privacy. For example, secure multi-party computation schemes enable two or more parties to compute some joint function while revealing no information about their respective inputs except what is leaked by the result of the computation. Privacy-preserving data mining aims to completely hide individual records while computing global statistical properties of the database. Search on encrypted data and private information retrieval enable the user to retrieve data from an untrusted server without revealing the query.

In this chapter, we investigate a different concept of privacy. Consider a data owner who wants to distribute a database to potential users. Instead of hiding individual data entries, he wants to obfuscate the database so that

only certain queries can be evaluated on it, *i.e.*, the goal is to ensure that the database, after it has been given out to users, can be accessed only in the ways permitted by the privacy policy. Note that there is no interaction between the data owner and the user when the latter accesses the obfuscated database.

Our constructions show how to obfuscate the database before distributing it to users so that only the queries permitted by the policy are computationally feasible. This concept of privacy is incomparable to conventional definitions because, depending on the policy, a permitted query may or even *should* reveal individual data entries.

For example, a college alumni directory may be obfuscated in such a way that someone who already knows a person's name and year of graduation is able to look up that person's email address, yet spammers cannot indiscriminately harvest addresses listed in the directory. Employees of a credit bureau need to have access to customers' records so that they can respond to reports of fraudulent transactions, yet one may want to restrict the bureau's ability to compile a list of customers' addresses and sell it to a third party.

We develop provably secure obfuscation techniques for several types of queries. We do *not* assume that users of the obfuscated database access it through a trusted third party, nor that they use trusted or "tamper-proof" access-control software or hardware (in practice, such schemes are vulnerable to circumvention and reverse-engineering, while trusted third parties are scarce and often impractical). Our constructions are *cryptographically* strong, *i.e.*, they assume an adversary who is limited only by his computational power.

We prove security in the standard "virtual black-box" model for obfuscation proposed by Barak *et al.* [27]. Intuitively, a database is securely obfuscated if the view of any efficient adversary with access to the obfuscation

can be efficiently simulated by a simulator who has access only to the *ideal functionality*, which is secure by definition. The ideal functionality can be thought of as the desired *privacy policy* for the database. One of our contributions is coming up with several ideal functionalities that capture interesting privacy policies for databases.

Directed-access databases. Our "warm-up" construction is a *directed-access database*. Some attributes of the database are designated as *query attributes*, and the rest as *data attributes*. The database is securely obfuscated if, for any record, it is infeasible to retrieve the values of the data attributes without supplying the values of the query attributes, yet a user who knows the query attributes can easily retrieve the corresponding data attributes.

To illustrate by example, a directed-access obfuscation of a telephone directory has the property that it is easy to look up the phone number corresponding to a particular name-address pair, but queries such as "retrieve all phone numbers stored in the directory" or "retrieve all names" are computationally infeasible. Such a directory is secure against abusive harvesting, but still provides useful functionality. Note that it may be possible to efficiently enumerate all name-address pairs because these fields have less entropy than regular cryptographic keys, and thus learn the entire database through the permitted queries. Because the database is accessed only in permitted ways, this does not violate the standard definition of obfuscation. Below, we give some examples where it is *not* feasible to enumerate all possible values for query attributes.

The directed-access property of a single database record can be modeled as a point function, *i.e.*, a function from $\{0,1\}^n$ to $\{0,1\}$ that returns 1 on

exactly one input x (in our case, query attributes are the arguments of the point function). Directed-access obfuscation guarantees that the adversary's view of any obfuscated record can be efficiently simulated with access only to this point function. Therefore, for this "warm-up" problem, we can use obfuscation techniques for point functions such as [170]. Informally, we encrypt the data attributes with a key derived from hashed query attributes. The only computationally feasible way to retrieve the data attributes is to supply the corresponding query attributes. If the retriever does not know the right query attributes, no information can be extracted at all.

Group-exponential databases. We then consider a more interesting privacy policy, which requires that computational cost of access be exponential in the number of database records retrieved. We refer to this new concept of privacy as *group privacy*. It ensures that users of the obfuscated database can retrieve individual records or small subsets of records by identifying them *precisely*, *i.e.*, by submitting queries which are satisfied *only* by these records. Queries matching a large number of records are infeasible.

We generalize the idea of directed access to queries consisting of conjunctions of equality tests on query attributes, and then to any boolean circuit over attribute equalities. The user can evaluate any query of the form $attribute_{j_1} = value_1 \ \wedge \ldots \wedge attribute_{j_t} = value_t$, as long as it is satisfied by a *small* number of records. Our construction is significantly more general than simple keyword search on encrypted data because the value of any query attribute or a conjunction thereof can be used as the "keyword" for searching the obfuscated database, and the obfuscator does not need to know what queries will be evaluated on the database.

To distinguish between "small" and "large" queries, suppose the query predicate is satisfied by n records. Our construction uses a form of secret sharing that forces the retriever to guess n bits before he can access the data attributes in any matching record. (If n=1, *i.e.*, the record is unique, the retriever still has to guess 1 bit, but this simply means that with probability $\frac{1}{2}$ he has to repeat the query.) The policy that requires the retriever to uniquely identify a single record, *i.e.*, forbids any query that is satisfied by multiple records, can also be easily implemented using our techniques. Our construction can be viewed as the non-interactive analog of hash-reversal "client puzzles" used to prevent denial of service in network security [142], but, unlike client puzzles, it comes with a rigorous proof of security.

For example, consider an airline passenger database in which every record contains the passenger's name, flight number, date, and ticket purchase details. In our construction, if the retriever knows the name and date that uniquely identify a particular record (*e.g.*, because this information was supplied in a court-issued warrant), he (almost) immediately learns the key that encrypts the purchase details in the obfuscated record. If the passenger traveled on k flights on that date, the retriever learns the key except for k bits. Since k is small, guessing k bits is still feasible. If, however, the retriever only knows the date and the flight number, he learns the key except for m bits, where m is the number of passengers on the flight, and retrieval of these passengers' purchase details is infeasible.

A database obfuscated using our method has the *group privacy* property in the following sense. It can be accessed only via queries permitted by the privacy policy. The probability of successfully evaluating a permitted query is inversely exponential in the number of records that satisfy the query predicate.

In particular, to extract a large number of records from the database, the retriever must know *a priori* specific information that uniquely identifies each of the records, or small subsets thereof. The obfuscated database itself does not help him obtain this information.

In obfuscated databases with group privacy, computational cost of access depends on the amount of information retrieved. Therefore, group privacy can be thought of as a step towards a formal cryptographic model for "economics of privacy." It is complementary to the existing concepts of privacy, and appears to be a good fit for applications such as public directories and customer relationship management (CRM) databases, where the database user may need to access an individual record for a legitimate business purpose, but should be prevented from extracting large subsets of records for resale and abusive marketing.

While our constructions for group privacy are provably secure in the "virtual black-box" sense of [27], the cost of this rigorous security is a quadratic blowup in the size of the obfuscated database, rendering the technique impractical for large datasets. We also present some heuristic techniques to decrease the size of the obfuscated database, and believe that further progress in this area is possible.

Alternative privacy policies. Defining rigorous privacy policies that capture intuitive "database privacy" is an important challenge, and we hope that this work will serve as a starting point in the discussion. For example, the group privacy policy that we use in our constructions permits the retriever to learn whether a given attribute of a database record is equal to a particular value. While this leaks more information than may be desirable, we conjecture

that the privacy policy without this oracle is *unrealizable*.

We also consider privacy policies that permit *any* query rather than just boolean circuits of equality tests on attributes. We show that this policy is *vacuous*: regardless of the database contents, any user can efficiently extract the entire database by policy-compliant queries. Therefore, even if the obfuscation satisfies the virtual black-box property, it serves no useful purpose. Of course, there are many types of queries that are more general than boolean circuits of equality tests on attributes. Exact characterization of non-vacuous, yet realizable privacy policies is a challenging task, and a topic of future research.

Organization of this chapter. We discuss related work in Section 5.2. The ideas are illustrated with a "warm-up" construction in Section 5.3. In Section 5.4, we explain group privacy and the corresponding obfuscation technique. In Section 5.5, we generalize the class of queries to boolean circuits over attribute equalities. In Section 5.6, we show that policies which permit arbitrary queries are vacuous, and give an informal argument that a policy that does not allow the retriever to verify his guesses of individual attribute values cannot be realized. A summary is in Section 5.7.

5.2 Related work

This work uses the "virtual black-box" model of obfuscation due to Barak *et al.* [27]. In addition to the impossibility result for general-purpose obfuscation, [27] demonstrates several classes of circuits that cannot be obfuscated. We focus on a different class of circuits.

To the best of our knowledge, the first provably secure constructions

for "virtual black-box" obfuscation were proposed by Canetti *et el.* [52, 53] in the context of "perfectly one-way" hash functions, which can be viewed as obfuscators for point functions (a.k.a. oracle indicators or delta functions). Dodis and Smith [78] recently showed how to construct noise-tolerant "perfectly one-way" hash functions. which they used to obfuscate proximity queries with "entropic security." It is not clear how to apply techniques of [78] in our setting. In section 5.6, we present strong evidence that our privacy definitions may not be realizable if queries other than equality tests are permitted.

Lynn *et al.* [170] construct obfuscators for point functions (and simple extensions, such as public regular expressions with point functions as symbols) in the random oracle model. The main advantage of [170] is that it allows the adversary partial information about the preimage of the hash function, *i.e.*, secrets do not need to have high entropy. This feature is essential in our constructions, too, thus we also use the random oracle model. Wee [256] proposed a construction for a weaker notion of point function obfuscation, along with the impossibility result for uniformly black-box obfuscation. This impossibility result suggests that the use of random oracles in our proofs (in particular, the simulator's ability to choose the random oracle) is essential.

Many ad-hoc obfuscation schemes have been proposed in the literature [21, 61, 60, 65, 66, 64]. Typically, these schemes contain neither a cryptographic definition of security, nor proofs, except for theoretical work on software protection with hardware restrictions on the adversary [115, 138].

Forcing the adversary to pay some computational cost for accessing a resource is a well-known technique for preventing malicious resource exhaustion (a.k.a. denial of service attacks). This approach, usually in the form of presenting a *puzzle* to the adversary and forcing him to solve it, has been

proposed for combating junk email [84], website metering [101], prevention of TCP SYN flooding attacks [142], protecting Web protocols [73], and many other applications. Puzzles based on hash reversal, where the adversary must discover the preimage of a given hash value where he already knows some of the bits, are an especially popular technique [142, 73, 255], albeit without any proof of security. Our techniques are similar, but our task is substantially harder in the context of *non-interactive* obfuscation.

The obfuscation problem is superficially similar to the problem of private information retrieval [29, 59, 112] and keyword search on encrypted data [232, 39]. These techniques are concerned, however, with retrieving data from an untrusted server, whereas we are concerned with encrypting the data and then giving them away, while preserving some control over what users can do with them.

A paper by Chawla *et al.* [56] also considers database privacy in a non-interactive setting, but their objective is complementary to ours. Their definitions aim to capture privacy of *data*, while ours aim to make *access* to the database indistinguishable from access to a certain ideal functionality.

5.3 Directed-access databases

As a warm-up example, we show how to construct *directed-access* databases in which every record is indistinguishable from a lookup function. The constructions and theorems in this section are mainly intended to illustrate the ideas.

Let $\mathcal{X}$ be a set of tuples $\overrightarrow{x}$, $\mathcal{Y}$ a set of tuples $\overrightarrow{y}$, and $\mathcal{Y}^* = \mathcal{Y} \cup \{\perp\}$. Let $\mathsf{D} \subseteq \mathcal{X} \times \mathcal{Y}$ be the database. We want to obfuscate each record of D so

that the *only* operation that a user can perform on it is to retrieve $\overrightarrow{y}$ if he knows $\overrightarrow{x}$.

We use the standard approach in secure multi-party computation, and formally define this privacy policy in terms of an *ideal functionality*. The ideal functionality is an (imaginary) trusted third party that permits only policy-compliant database accesses. An obfuscation algorithm is secure if any access to the obfuscated database can be efficiently simulated with access only to the ideal functionality. This means that the user can extract no more information from the obfuscated database than he would be able to extract had all of his accesses been filtered by the trusted third party.

Definition 1 (Directed-access privacy policy) *For database* D*, define the corresponding* directed-access functionality $\mathcal{DA}_{\mathsf{D}}$ *as the function that, for any input* $\overrightarrow{x} \in \mathcal{X}$ *such that* $\langle \overrightarrow{x}, \overrightarrow{y_1} \rangle, \ldots, \langle \overrightarrow{x}, \overrightarrow{y_m} \rangle \in \mathsf{D}$*, outputs* $\{\overrightarrow{y_1}, \ldots, \overrightarrow{y_m}\}$.

Intuitively, a directed-access database is indistinguishable from a lookup function. Given the query attributes of an individual record ($\overrightarrow{x}$), it is easy to learn the data attributes ($\overrightarrow{y}$), but the database cannot be feasibly accessed in any other way. In particular, it is not feasible to discover the value of $\overrightarrow{y}$ without first discovering a corresponding $\overrightarrow{x}$. Moreover, it is not feasible to harvest all $\overrightarrow{y}$ values from the database without first discovering *all* values of $\overrightarrow{x}$.

This definition does *not* say that, if set $\mathcal{X}$ is small, it is infeasible to efficiently enumerate all possible values of $\overrightarrow{x}$ and stage a dictionary attack on the obfuscated database. It *does* guarantee that even for this attack, the attacker is unable to evaluate any query forbidden by the privacy policy. In applications where $\mathcal{X}$ cannot be efficiently enumerated (*e.g.*, $\mathcal{X}$ is a set of

secret keywords known only to some users of the obfuscated database), nothing can be retrieved from the obfuscated database by users who don't know the keywords. Observe that $\overrightarrow{x}$ can contain multiple attributes, and thus multiple keywords may be required for access to $\overrightarrow{y}$ in the obfuscated database.

Directed-access databases are easy to construct in the random oracle model, since lookup functionality is essentially a point function on query attributes, and random oracles naturally provide an obfuscation for point functions [170]. The obfuscation algorithm $\mathcal{OB}_{da}$ takes D and replaces every record $\langle \overrightarrow{x_i}, \overrightarrow{y_i} \rangle \in \mathsf{D}$ with

$$\langle \mathsf{hash}(r_{i_1} || \overrightarrow{x_i}), \mathsf{hash}(r_{i_2} || \overrightarrow{x_i}) \oplus \overrightarrow{y_i}, r_{i_1}, r_{i_2} \rangle$$

where $r_{i_{1,2}}$ are random numbers, $||$ is concatenation, and hash is a hash function implementing the random oracle.

Theorem 1 (Directed-access obfuscation is "virtual black-box") *Let $\mathcal{OB}_{da}$ be as described above. For any probabilistic polynomial-time adversarial algorithm A, there exists a probabilistic polynomial-time simulator algorithm S and a negligible function ν of the security parameter k such that for any database D:*

$$|\mathbf{P}(A(\mathcal{OB}_{da}(\mathsf{D})) = 1) - \mathbf{P}(S^{\mathcal{D},\mathcal{A}_{\mathsf{D}}}(1^{|\mathsf{D}|}) = 1)| \leq \nu(k)$$

where probability $\mathbf{P}$ is taken over random oracles (implemented as hash functions), as well as the the randomness of A and S. Intuitively, this theorem holds because retrieving $\overrightarrow{y_i}$ requires finding the (partial) pre-image of $\mathsf{hash}(r_{i_2}, \overrightarrow{x_i})$.

The standard definition of obfuscation in [27] also requires that there

exist an efficient retrieval algorithm that, given some $\overrightarrow{x}^*$, extracts the corresponding $\overrightarrow{y}$ from the obfuscation $\mathcal{OB}_{da}(\mathsf{D})$. Clearly, our construction has this property. Someone who knows $\overrightarrow{x}^*$ simply finds the record(s) in which the first value is equal to $\mathsf{hash}(r_{i_1}||\overrightarrow{x}^*)$, computes $\mathsf{hash}(r_{i_2}||\overrightarrow{x}^*)$ and uses it as the key to decrypt $\overrightarrow{y}$.

5.4 Group-exponential databases

For the purposes of this section, we restrict our attention to queries $\mathcal{P}$ that are conjunctions of equality tests over attributes (in section 5.5, we show how this extends to arbitrary boolean circuits over equality tests). For this class of queries, we show how to obfuscate the database so that evaluation of the query is exponential in the *size of the answer to the query*. Intuitively, this means that only precise query predicates, *i.e.*, those that are satisfied by a small number of records, can be efficiently computed. "Mass harvesting" queries, *i.e.*, predicates that are satisfied by a large number of records, are computationally infeasible.

Recall that our goal is to restrict *how* the database can be accessed. For some databases, it may be possible to efficiently enumerate all possible combinations of query attributes and learn the entire database by querying it on every combination. For databases where the values of query attributes are drawn from a large set, our construction prevents the retriever from extracting any records that he cannot describe precisely. In either case, we guarantee that the database can be accessed *only* through the interface permitted by the privacy policy, without any trust assumptions about the retriever's computing environment.

In our construction, each data attribute is encrypted with a key derived from a randomly generated secret. We use a different secret for each record. The secret itself is split into several (unequal) shares, one per each query attribute. Each share is then encrypted itself, using a key derived from the output of the hash function on the value of the corresponding query attribute. If the retriever knows the correct value only for some query attribute a, he must guess the missing shares. The size of the revealed share in bits is inversely related to the number of other records in the database that have the same value of attribute a. This provides protection against queries on frequently occurring attribute values.

5.4.1 Group privacy policy

We define the privacy policy in terms of an *ideal functionality*, which consists of two parts. When given an index of a particular query attribute and a candidate value, it responds whether the guess is correct, *i.e.*, whether this value indeed appears in the corresponding attribute of the original database. When given a predicate, it evaluates this predicate on every record in the database. For each record on which the predicate is true, it returns this record's data attributes with probability 2^{-q}, where q is the total number of records in the database that satisfy the predicate. if no more information can be extracted this ideal functionality.

Definition 2 (GROUP PRIVACY POLICY) *Let $\mathcal{X}$ be a set and $\mathcal{Y}$ a set of tuples. Let D be the database $\langle \rho_1, \rho_2, \ldots \rho_N \rangle$ where $\rho_i = \{x_{i1}, x_{i2}, \ldots, x_{im}, \overrightarrow{y_i}\} \in \mathcal{X}^m \times \mathcal{Y}$. Let $\mathcal{P} : \mathcal{X}^m \rightarrow \{0,1\}$ be a predicate of the form $X_{j_1} = x_{j_1} \wedge X_{j_2} = x_{j_2} \wedge \ldots \wedge X_{j_t} = x_{j_t}$. Let $\mathsf{D}_{[\mathcal{P}]} = \{\rho_i \in \mathsf{D} \mid \mathcal{P}(x_{i1}, x_{i2}, \ldots, x_{im}) = 1\}$ be the*

subset of records on which $\mathcal{P}$ is true.

The group-exponential functionality $\mathcal{GP}_{\mathsf{D}}$ *consists of two functions:*
- $C_{\mathsf{D}}(x, i, j)$ *is 1 if* $x = x_{ij}$ *and 0 otherwise, where* $1 \leq i \leq N, 1 \leq j \leq m$.
- $R_{\mathsf{D}}(\mathcal{P}) = \bigcup_{1 \leq i \leq N} \{\langle i, \gamma_i \rangle\}$, *where*

$$\gamma_i = \begin{cases} \overrightarrow{y_i} & \textit{with probability } 2^{-|\mathsf{D}_{[\mathcal{P}]}|} \textit{ if } \mathcal{P}(\rho_i) \\ \perp & \textit{with probability } 1 - 2^{-|\mathsf{D}_{[\mathcal{P}]}|} \textit{ if } \mathcal{P}(\rho_i) \\ \perp & \textit{if } \neg\mathcal{P}(\rho_i) \end{cases}$$

Probability is taken over the internal coin tosses of $\mathcal{GP}_{\mathsf{D}}$.

Informally, function C answers whether the jth attribute of the ith record is equal to x, while function R returns all records that satisfy some predicate $\mathcal{P}$, but only with probability inversely exponential in the number of such records.

It may appear that function C is unnecessary. Moreover, it leaks additional information, making our privacy policy weaker than it might have been. In section 5.6, we argue that it cannot be simply eliminated, because the resulting functionality would be *unrealizable*. Of course, there may exist policies that permit some function C' which leaks less information than C, but it is unclear what C' might be. We discuss several alternatives to our definition in section 5.6.

We note that data attributes are drawn from a set of tuples $\mathcal{Y}$ because there may be multiple data attributes that need to be obfuscated. Also observe that we have no restrictions on the values of query attributes, *i.e.*, the same m-tuple of query attributes may appear in multiple records, with different or identical data attributes.

5.4.2 Obfuscating the database

We now present the algorithm $\mathcal{OB}_{gp}$, which, given any database D, produces its obfuscation. For notational convenience, we use a set of random hash functions $H_\alpha : \{0,1\}^* \rightarrow \{0,1\}^k$. Given any hash function H, these can be implemented simply as $H(\alpha||x)$. To convert the k-bit hash function output into a key as long as the plaintext to which it is applied, we use a set of pseudo-random number generators $\mathsf{prg}_{\alpha,\beta} : \{0,1\}^k \rightarrow \{0,1\}^\infty$ (this implements random oracles with unbounded output length).

Let N be the number of records in the database. For each row ρ_i, $1 \leq i \leq N$, generate a random N-bit secret $\mathbf{r}_i = r_{i1}||r_{i2}||\ldots||r_{iN}$, where $r_{ij} \in_R \{0,1\}$. This secret will be used to protect the data attribute $\overrightarrow{y_i}$ of this record. Note that there is 1 bit in $\mathbf{r}_i$ for each record of the database.

Next, split $\mathbf{r}_i$ into m shares corresponding to query attributes. If the retriever can supply the correct value of the jth attribute, he will learn the jth share ($1 \leq j \leq m$). Denote the share corresponding to the jth attribute as s_{ij}. Shares are also N bits long, *i.e.*, $s_{ij} = s_{ij1}||\ldots||s_{ijN}$.

Each of the N bits of s_{ij} has a corresponding bit in $\mathbf{r}_i$, which in its turn corresponds to one of the N records in the database. For each p s.t. $1 \leq p \leq N$, set $s_{ijp} = \mathbf{r}_{ip}$ if $x_{ij} \neq x_{pj}$, and set $s_{ijp} = 0$ otherwise. In other words, the jth share s_{ij} consists of all bits of $\mathbf{r}_i$ *except* those corresponding to the records where the value of the jth attribute is the same. An example can be found in section 5.4.4.

The result of this construction is that shares corresponding to commonly occurring attribute values will be missing many bits of $\mathbf{r}_i$, while a share corresponding to an attribute that uniquely identifies one record will contain

all bits of $\mathbf{r}_i$ except one. Intuitively, this guarantees group privacy. If the retriever can supply query attribute values that uniquely identify a single record or a small subset of records, he will learn the shares that reveal all bits of the secret $\mathbf{r}_i$ except for a few, which he can easily guess. If the retriever cannot describe precisely what he is looking for and supplies attribute values that are common in the database, many of the bits of $\mathbf{r}_i$ will be missing in the shares that he learns, and guessing all of them will be computationally infeasible.

Shares corresponding to different query attributes may overlap. For example, suppose that we are obfuscating a database in which two records have the same value of attribute X_1 if and only if they have the same value of attribute X_2. In this case, for any record in the database, the share revealed if the retriever supplies the correct value of X_1 will be exactly the same as the share revealed if the retriever supplies the value of X_2. The retriver gains nothing by supplying X_2 in conjunction with X_1 because this does not help him narrow the set of records that match his query.

To construct the obfuscated database, we encrypt each share with a pseudo-random key derived from the value of the corresponding query attribute, and encrypt the data attribute with a key derived from the secret $\mathbf{r}_i$. More precisely, we replace each record $\langle \rho_i = x_{i1}, \ldots, x_{im}, \overrightarrow{y_i} \rangle$ of the original database with the obfuscated record

$$\langle v_{i1}, w_{i1}, v_{i2}, w_{i2}, \ldots, v_{im}, w_{im}, u_i, z_i \rangle$$

where

- $v_{ij} = H_{1,i,j}(x_{ij})$. This enables the retriever to verify that he supplied the correct value for the jth query attribute.

- $w_{ij} = \mathsf{prg}_{1,i,j}(H_{2,i,j}(x_{ij})) \oplus s_{ij}$. This is the jth share of the secret $\mathbf{r}_i$, encrypted with a key derived from the value of the jth query attribute.

- $u_i = H_{3,i}(\mathbf{r}_i)$. This enables the retriever to verify that he computed the correct secret $\mathbf{r}_i$.

- $z_i = \mathsf{prg}_{2,i}(H_{4,i}(\mathbf{r}_i)) \oplus \overrightarrow{y_i}$. This is the data attribute $\overrightarrow{y_i}$, encrypted with a key derived from the secret $\mathbf{r}_i$.

Clearly, algorithm $\mathcal{OB}_{gp}$ runs in time polynomial in N (the size of the database). The size of the resulting obfuscation is $N^2 \cdot m$. Even though it is within a polynomial factor of N (and thus satisfies the definition of [27]), quadratic blowup means that our technique is impractical for large databases. This issue is discussed further in section 5.4.5.

We claim that $\mathcal{OB}_{gp}$ produces a secure obfuscation of D, *i.e.*, it is not feasible to extract any more information from $\mathcal{OB}_{gp}(\mathsf{D})$ than permitted by the privacy policy $\mathcal{GP}_{\mathsf{D}}$.

Theorem 2 (Group-exponential obfuscation is "virtual black-box") *For any probabilistic polynomial-time (adversarial) algorithm A, there exists a probabilistic polynomial-time simulator algorithm S and a negligible function ν of the security parameter k s.t. for any database D:*

$$|\mathbf{P}(A(\mathcal{OB}_{gp}(\mathsf{D})) = 1) - \mathbf{P}(S^{\mathcal{GP}_{\mathsf{D}}}(1^{|\mathsf{D}|}) = 1)| \leq \nu(k)$$

Remark. An improper implementation of the random oracles in the above construction could violate privacy under composition of obfuscation, *i.e.*, when

more than one database is obfuscated and published. For instance, if the hash of some attribute is the same in two databases, then the adversary learns that the attributes are equal without having to guess their value. To prevent this, the same hash function may not be used more than once. One way to achieve this is to pick $H_i(.) = H(r_i||.)$ where $r_i \in_R \{0,1\}^k$, and publish r_i along with the obfuscation. This is an example of the pitfalls inherent in the random oracle model.

5.4.3 Accessing the obfuscated database

We now explain how the retriever can efficiently evaluate queries on the obfuscated database. Recall that the privacy policy restricts the retriever to queries consisting of conjunctions of equality tests on query attributes, *i.e.*, every query predicate $\mathcal{P}$ has the form $X_{j_1} = x_{j_1} \wedge \ldots \wedge X_{j_t} = x_{j_t}$, where $j_1, \ldots, j_t$ are some indices between 1 and m.

The retriever processes the obfuscated database record by record. The ith record of the obfuscated database ($1 \leq i \leq N$) has the form $\langle v_{i1}, w_{i1}, v_{i2}, w_{i2}, \ldots, v_{im}, w_{im}, u_i, z_i \rangle$ The retriever's goal is to compute the N-bit secret $\mathbf{r}_i$ so that he can decrypt the ciphertext z_i and recover the value of $\overrightarrow{y_i}$.

First, the retriever recovers as many shares as he can from the ith record. Recall from the construction of section 5.4.2 that each w_{ij} is a ciphertext of some share, but the only way to decrypt it is to supply the corresponding query attribute value x_{ij}. Let ℓ range over the indices of attributes supplied by the retriever as part of the query, *i.e.*, $\ell \in \{j_1, \ldots, j_t\}$. For each ℓ, if $H_{1,i,\ell}(x_\ell) = v_{i\ell}$, then the retriever extracts the corresponding share $s_{i\ell} = \mathsf{prg}_{1,i,\ell}(H_{2,i,\ell}(x_\ell)) \oplus w_{i\ell}$. If $H_{1,i,\ell}(x_\ell) \neq v_{i\ell}$, this means that the

retriever supplied the wrong attribute value, and he learns nothing about the corresponding share. Let S be the set of recovered shares.

Each recovered share $s_\ell \in S$ reveals only some bits of $\mathbf{r}_i$, and, as mentioned before, bits revealed by different shares may overlap. For each p s.t. $1 \leq p \leq N$, the retriever sets the corresponding bit $\mathbf{r}_{ip}$ of the candidate secret $\mathbf{r}_i$ as follows:

$$\mathbf{r}_{ip} = \begin{cases} s_{\ell p} & \text{if } \exists s_\ell \in S \text{ s.t. } v_{p\ell} \neq H_{1,1,\ell}(x_\ell) \\ \text{random} & \text{otherwise} \end{cases}$$

Informally, if at least one of recovered shares s_ℓ contains the pth bit of $\mathbf{r}_i$ (this can be verified by checking that the value of ℓth attribute is *not* the same in the pth record of the database — see construction in section 5.4.2), then this bit is indeed to the pth bit of the secret $\mathbf{r}_i$. Otherwise, the retriever must guess the pth bit randomly.

Once a candidate $\mathbf{r}_i$ is constructed, the retriever checks whether $H_{3,i}(\mathbf{r}_i) = u_i$. If not, the missing bits must have been guessed incorrectly, and the retriever has to try another choice for these bits. If $H_{3,i}(\mathbf{r}_i) = u_i$, then the retriever decrypts the data attribute $\overrightarrow{y_i} = \mathsf{prg}_{2,i}(H_{4,i}(\mathbf{r}_i)) \oplus z_i$.

The obfuscation algorithm of section 5.4.2 guarantees that the number of missing bits is exactly equal to the number of records satisfied by the query $\mathcal{P}$. This provides the desired group privacy property. If the retriever supplies a query which is satisfied by a single record, then he will only have to guess one bit to decrypt the data attributes. If a query is satisfied by two records, then two bits must be guessed, and so on. For queries satisfied by a large number of records, the number of bits to guess will be infeasible large.

5.4.4 Example

Consider a toy airline passenger database with 4 records, where the query attributes are "Last name" and "Flight," and the data attribute (in bold) is "Purchase details."

Last name	Flight	**Purchase details**
Smith	88	**Acme Travel, Visa 4390XXXX**
Brown	500	**Airline counter, cash**
Jones	88	**Nonrevenue**
Smith	1492	**Travel.com, AmEx 3735XXXX**

Because $N = 4$, we need to create a 4-bit secret to protect each data attribute. (4-bit secrets can be easily guessed, of course. We assume that in real applications N would be sufficiently large, and use 4 records in this example only to simplify the explanations.) Let $\alpha = \alpha_1\alpha_2\alpha_3\alpha_4$ be the secret for the first data attribute, and β, γ, δ the secrets for the other data attributes, respectively.

For simplicity, we use a special symbol "?" to indicate the missing bits that the retriever must guess. In the actual construction, each of these bits is equal to 0, but the retriever knows that he must guess the ith bit of the jth share if the value of the jth attribute in the current record is equal to the value of the jth attribute in the ith record.

Consider the first record. Each of the two query attributes, "Last name" and "Flight," encrypts a 4-bit share. The share encrypted with the value of the "Last name" attribute (*i.e.*, "Smith") is missing the 1st and 4th bits because the 1st and 4th records in the database have the same value of this attribute. (Obviously, all shares associated the ith record have the ith bit missing). The

share encrypted with the value of the "Flight" attribute is missing the 1st and 3rd bits.

$H_{111}(\text{“Smith”}), \mathsf{prg}_{1,1,1}(H_{211}(\text{“Smith”})) \oplus (?\alpha_2\alpha_3?)$,
$H_{112}(\text{“88”}), \mathsf{prg}_{1,1,2}(H_{212}(\text{“88”})) \oplus (?\alpha_2?\alpha_4)$,
$H_{31}(\alpha_1\alpha_2\alpha_3\alpha_4), \mathsf{prg}_{2,1}(H_{41}(\alpha_1\alpha_2\alpha_3\alpha_4)) \oplus (\text{“Acme...”})$
$H_{121}(\text{“Brown”}), \mathsf{prg}_{1,2,1}(H_{221}(\text{“Brown”})) \oplus (\beta_1?\beta_3\beta_4)$,
$H_{122}(\text{“500”}), \mathsf{prg}_{1,2,2}(H_{222}(\text{“500”})) \oplus (\beta_1?\beta_3\beta_4)$,
$H_{32}(\beta_1\beta_2\beta_3\beta_4), \mathsf{prg}_{2,2}(H_{42}(\beta_1\beta_2\beta_3\beta_4)) \oplus (\text{“Airline...”})$
$H_{131}(\text{“Jones”}), \mathsf{prg}_{1,3,1}(H_{231}(\text{“Jones”})) \oplus (\gamma_1\gamma_2?\gamma_4)$,
$H_{132}(\text{“88”}), \mathsf{prg}_{1,3,2}(H_{232}(\text{“88”})) \oplus (?\gamma_2?\gamma_4)$,
$H_{33}(\gamma_1\gamma_2\gamma_3\gamma_4), \mathsf{prg}_{2,3}(H_{43}(\gamma_1\gamma_2\gamma_3\gamma_4)) \oplus (\text{“Nonrev...”})$
$H_{141}(\text{“Smith”}), \mathsf{prg}_{1,4,1}(H_{241}(\text{“Smith”})) \oplus (?\delta_2\delta_3?)$,
$H_{142}(\text{“1492”}), \mathsf{prg}_{1,4,2}(H_{242}(\text{“1492”})) \oplus (\delta_1\delta_2?\delta_4)$,
$H_{34}(\delta_1\delta_2\delta_3\delta_4), \mathsf{prg}_{2,4}(H_{44}(\delta_1\delta_2\delta_3\delta_4)) \oplus (\text{“Travel.com...”})$

Suppose the retriever knows only that the flight number is 88. There are 2 records in the database that match this predicate. From the first obfuscated record, he recovers $?\alpha_2?\alpha_4$ and from the third obfuscated record, $?\gamma_2?\gamma_4$. The retriever learns which bits he must guess by computing $H_{2i2}(\text{“88”})$ for $1 \leq i \leq 4$, and checking whether the result is equal to v_{i2} from the ith obfuscated record. In both cases, the retriever learns that he must guess 2 bits (1st and 3rd) in order to reconstruct the secret and decrypt the data attribute.

Now suppose the retriever knows that the flight number is 88 and the last name is Smith. There is only 1 record in the database that satisfies this predicate. From the first part of the first obfuscated record, the retriever

can recover $?\alpha_2\alpha_3?$, and from the second part $?\alpha_2?\alpha_4$ (note how the shares overlap). Combining them, he learns $?\alpha_2\alpha_3\alpha_4$, so he needs to guess only 1 bit to decrypt the data attribute.

It may appear that this toy example is potentially vulnerable to a dictionary attack, since it is conceivable that all combinations of last names and flight numbers can be efficiently enumerated with enough computing power. Note, however, that this "attack" does *not* violate the definition of secure obfuscation because the retriever must supply the name-flight pair before he can recover the purchase details. Therefore, the obfuscated database is only accessed via queries permitted by the privacy policy. In databases where values are drawn from a large set, even this "attack" is infeasible.

5.4.5 Efficiency

The algorithm of section 5.4.2 produces obfuscations which are a factor of $\Omega(N)$ larger than original databases. Thus, while our results establish feasibility of database obfuscation and group privacy, they are not directly applicable to real-world databases. This appears to be a recurring problem in the field of database privacy: the cryptography community has very strict definitions of security but loose notions of efficiency (typically polynomial time and space), whereas the database community has very strict efficiency requirements but loose security (typically heuristic or statistical). As a result, many proposed schemes are either too inefficient, or too insecure for practical use.

A possible compromise might be to start with a provably secure but inefficient construction and employ heuristic techniques to improve its efficiency. In this spirit, we now propose some modifications to reduce the size of

the obfuscated database without providing a security proof.

The obfuscation algorithm is modified as follows. For each record i, we split $\mathbf{r}_i$ into $\frac{N}{k}$ "blocks" of k bits each, padding the last block if necessary (k is the security parameter). Instead of generating the bits randomly, we create a binary tree of depth $\log \frac{N}{k}$. A key of length k is associated with each node of the tree, with the property the two "children" keys are derived from the "parent" key (*e.g.*, using a size-doubling pseudo-random generator). This is similar to a Merkle tree in which keys are derived in the *reverse* direction. The edge of the tree (minus the padding of the last block) is $\mathbf{r}_i$.

Let us denote the j^{th} attribute of the i^{th} record by $\langle i, j \rangle$. Say that $\langle i, j \rangle$ is *entitled* to the secret bit $r_{i'j}$ if $x_{ij} \neq x_{i'j}$, and $\langle i, j \rangle$ is entitled to an entire block B if it is entitled to each secret bit $r_{i'j}$ in that block. Intuitively, if an entire block is entitled, then we encode it efficiently using the "reverse Merkle" tree described above; if it is partially entitled, then we fall back on our original construction. Thus, let $\mathcal{N}_{ij}$ be the minimal set of nodes in the tree which are sufficient for reconstructing all entitled blocks (*i.e.*, every entitled block has among its parents an element of $\mathcal{N}_{ij}$), and only these blocks. Then the share s_{ij} consists of (a suitable encoding of) $\mathcal{N}_{ij}$ together with the remaining bits $r_{i'j}$ to which $\langle i, j \rangle$ is entitled. These are the entitled bits from any block which also includes non-entitled bits.

In the worst case, this algorithm does not decrease the blowup in the size of the obfuscated database. This occurs when for every query attribute j of every record i, there are $\Omega(N)$ records i' for which the value of the query attribute is the same, *i.e.*, $x_{ij} = x_{i'j}$. If we assume a less pathological database, however, we can get a better upper bound. If there is a threshold t such that for any (i, j) there are at most t records i' for which $x_{ij} = x_{i'j}$, then the size

of the key material (which causes the blowup in the size of the obfuscated database) is $O(mNt(k \log \frac{N}{k}))$ bits (recall that m is the number of attributes). This bound is tight only for small values of t, and the new algorithm does no worse than the original one even when $t = \Omega(N)$. When we consider that each of the mN entries of the original database is several bits long, the size of the obfuscated database could be acceptably small for practical use.

It must be noted that this obfuscation reveals the size of the share, and thus, for a given attribute of a given record, it leaks information about the *number* of other records whose attribute value is the same (but not *which* records they are). This opens two research questions:

- Is there a provably secure database obfuscation algorithm that produces obfuscations that are smaller than $O(N^2)$.
- Can the heuristic described in this section be improved to obtain acceptable lower bounds in the worst case?

5.5 Arbitrary predicates over equalities on attributes

We now consider queries formed by taking an arbitrary predicate $\mathcal{P}$ over m boolean variables $b_1, b_2 \ldots b_m$, and substituting $(X_j = x_j)$ for b_j, where X_j is a query attribute, and $x_j \in \mathcal{X} \cup \{*\}$ is a candidate value for this attribute, drawn from the domain $\mathcal{X}$ of query attribute values. The special value $*$ denotes that the value of the X_j attribute is ignored when evaluating the predicate. The class of queries considered in section 5.4 is a partial case of this

definition, where $\mathcal{P} = \bigwedge_{1 \leq j \leq m} b_j$. The group-exponential property is similar to definition 2 except for the domain of $\mathcal{P}$.

Let $\mathcal{C}$ be a boolean circuit computing $\mathcal{P}$. We assume that $\mathcal{C}$ is a *monotone* circuit, *i.e.*, a poly-size directed acyclic graph where each node is an AND, OR or FANOUT gate. AND and OR gates have two inputs and one output each, while FANOUT gates have one input and two outputs. Circuit $\mathcal{C}$ has m inputs, one per each query attribute. Below, we show how to generalize our obfuscation technique to non-monotone circuits.

Obfuscation algorithm. The algorithm is similar to the one in section 5.4, and consists of generating a random secret to encrypt each data attribute, splitting this secret into (unequal) shares, and encrypting these shares under the keys derived from the values of query attributes.

As before, let $H_\alpha : \{0,1\}^* \rightarrow \{0,1\}^k$ be a set of random hash functions and $\mathsf{prg}_{\alpha,\beta} : \{0,1\}^k \rightarrow \{0,1\}^\infty$ a set of pseudo-random generators.

For each record ρ_i in the database, do the following:

- Generate a block of uniformly random bits $\{r_{ilEt}\}$, where $1 \leq l \leq N$, E ranges over all edges of the circuit $\mathcal{C}$, and $1 \leq t \leq k$, where k is the length of the hash functions' output. Denote

$$\mathbf{r}_{iEt} = r_{i1Et}||r_{i2Et}|| \ldots ||r_{iNEt}$$
$$\overrightarrow{r_{ilE}} = r_{ilE1}||r_{ilE2}|| \ldots ||r_{ilEk}$$

- Then, for each query attribute X_j:
  - Output $v_{ij} = H_{1,i,j}(x_{ij})$
  - Let E_j be the input edge in the circuit $\mathcal{C}$ whose input is the $X_j = x_j$

test. Define the bits of the corresponding share $s_{iljt} = r_{ilE_jt}$ if $x_{ij} \neq x_{lj}$, and 0 otherwise. Encrypt the resulting share using a key derived from x_{ij}, *i.e.*, output

$$w_{ij} = \mathsf{prg}_{1,i,j}(H_{2,i,j}(x_{ij})) \oplus (\overrightarrow{s_{i1j}}||\overrightarrow{s_{i2j}}|| \ldots ||\overrightarrow{s_{iNj}}).$$

- Let E_{out} be the output edge in the circuit $\mathcal{C}$. Output $u_i = H_{3,i}(\mathbf{r}_{iE_{out}0})$
- Output $z_i = \mathsf{prg}_{2,i}(H_{4,i}(\mathbf{r}_{iE_{out}0})) \oplus \overrightarrow{y_i}$.
- The previous procedure obfuscated only the output edge of $\mathcal{C}$. Repeat the following step recursively for each gate $\mathcal{G} \in \mathcal{C}$, whose output edge (or both of whose output edges, for a FANOUT gate) have been obfuscated. Stop when all edges have been obfuscated:
  - If $\mathcal{G}$ is an AND gate, let E_0 and E_1 be the input edges and E the output edge. For each l, set $\overrightarrow{r_{ilE_0}} = \overrightarrow{r_{ilE_1}} = \overrightarrow{r_{ilE}}$.
  - If $\mathcal{G}$ is an OR gate, then, for each l, generate random $\overrightarrow{r_{ilE_0}} \in_R \{0,1\}^k$ and set $\overrightarrow{r_{ilE_1}} = \overrightarrow{r_{ilE_0}} \oplus \overrightarrow{r_{ilE}}$.
  - If $\mathcal{G}$ is a FANOUT gate, let E_0 and E_1 be the output edges and E the input edge. For each l, generate random $\overrightarrow{r_{ilE_0}}, \overrightarrow{r_{ilE_1}} \in_R \{0,1\}^k$ and output

    $$\zeta_{ilE_0} = H_{5,i,l,E_0}(\overrightarrow{r_{ilE}}) \oplus \overrightarrow{r_{ilE_0}}$$

    and

    $$\zeta_{ilE_1} = H_{5,i,l,E_1}(\overrightarrow{r_{ilE}}) \oplus \overrightarrow{r_{ilE_1}}$$

Retrieval algorithm. Let $\mathcal{Q}$ be the query predicate in which specific values

of x_j or $*$ have been plugged into all $X_j = x_j$ expressions in the leaves of the circuit $\mathcal{C}$.

The retrieval algorithm consists of two functions:
$C_{ob}(\mathcal{OB}_{gp}(\mathsf{D}), x, i, j)$, which enables the retriever to check whether the jth query attribute of the ith record is equal to x, and $R_{ob}(\mathcal{OB}_{gp}(\mathsf{D}), \mathcal{Q}, i)$, which attempts to retrieve the value of the obfuscated data attribute in the ith record.

Define $C_{ob}(\mathcal{OB}_{gp}(\mathsf{D}), x, i, j) = 1$ if $H_{1,i,j}(x) = v_{ij}$ and 0 otherwise.

- Evaluate $\mathcal{Q}(\rho_i)$ using C_{ob}. If $\neg\mathcal{Q}_{\mathcal{OB}_{gp}}(\rho_i)$, then $R_{ob}(\mathcal{OB}_{gp}(\mathsf{D}), \mathcal{Q}, i) = \perp$.

- For each l and each circuit edge E, set $\overrightarrow{r_{ilE}} = ?? \ldots ?$ (*i.e.*, none of the bits of the secret are initially known).

- For each query attribute j, let E_j be the input edge of the circuit associated with the equality test for this attribute. If $\mathcal{Q}$ contains this test, *i.e.*, if $\mathcal{Q}$ contains $X_j = x_j$ for some candidate value x_j (rather than $X_j = *$), then set $(\overrightarrow{s_{i1j}}|| \ldots ||\overrightarrow{s_{iNj}}) = w_{ij} \oplus \mathsf{prg}_{1,i,j}(H_{2,i,j}(x_{ij}))$, *i.e.*, decrypt the secret bits with the key derived from the value of the jth attribute.

  For each l, if $C_{ob}(x_{ij}, l, j) = 0$, then set $\overrightarrow{r_{ilE_j}} = \overrightarrow{s_{ilj}}$, *i.e.*, use only those of the decrypted bits that are true bits of the secret $\overrightarrow{r_{ilE}}$.

- So far, only the input gates of the circuit have been visited. Find a gate all of whose input edges have been visited, and repeat the following step for every gate until the output edge E_{out} has been visited.

  - If E is the output of an AND gate with inputs E_0 and E_1, then, for each l, if $\overrightarrow{r_{ilE_0}} \neq ?$, set $\overrightarrow{r_{ilE}} = \overrightarrow{r_{ilE_0}}$; if $\overrightarrow{r_{ilE_1}} \neq ?$, set $\overrightarrow{r_{ilE}} = \overrightarrow{r_{ilE_1}}$.

- E is the output of an OR gate with inputs E_0 and E_1. For each l, if $\overrightarrow{r_{ilE_0}} \neq ?$ and $\overrightarrow{r_{ilE_1}} \neq ?$, set $\overrightarrow{r_{ilE}} = \overrightarrow{r_{ilE_0}} \oplus \overrightarrow{r_{ilE_1}}$.
- E is the output of a FANOUT gate with input E_0. For each l, if $\overrightarrow{r_{ilE_0}} \neq ?$, set $r_{ilE} = \zeta_{ilE_0} \oplus H_{5,i,l,E_0}(\overrightarrow{r_{ilE_0}})$.

- For each l, if $r_{ilE_{out}0} = ?$, this means that the corresponding secret bit must be guessed. Choose random $r_{ilE_{out}0} \in_R \{0,1\}$.

- If $H_{3,i}(\mathbf{r}_{iE_{out}0}) = u_i$, this means that the retriever successfully reconstructed the secret. In this case, define $R_{ob}(\mathcal{OB}_{gp}(\mathsf{D}), \mathcal{Q}, i) = \mathsf{prg}_{2,i}(H_{4,i}(r_{iE_{out}0})) \oplus z_i$. Otherwise, define $R_{ob}(\mathcal{OB}_{gp}(\mathsf{D}), \mathcal{Q}, i) = \perp$.

Theorem 3 *The obfuscation algorithm for arbitrary predicates over equalities on attributes satisfies the virtual black-box property.*

5.5.1 Obfuscating non-monotone circuits

Given a non-monotone circuit $\mathcal{C}$, let $\underline{\mathcal{C}}$ be the monotone circuit whose leaves are literals and negated literals formed by "pushing down" all the NOT gates. Observe that $\underline{\mathcal{C}}$ has at most twice as many gates as $\mathcal{C}$. Also, $\underline{\mathcal{C}}$ can be considered a monotone circuit over the $2m$ predicates $X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m, X_1 \neq x_1, X_2 \neq x_2, \ldots X_m \neq x_m$. Observe that a predicate of the form $X_j \neq x_j$ is meaningful only when $x_j = x_{ij}$ for some record i. This is because if $x_j \neq x_{ij}$ for any record i, then $X_j \neq x_j$ matches *all* the records. Hence there exists a circuit $\underline{\mathcal{C}}'$ (obtained by setting the leaf in $\underline{\mathcal{C}}$ corresponding to the predicate $X_j \neq x_j$ to `true`) that evaluates to the same value as $\underline{\mathcal{C}}$ for every record in the database.

Given that $x_j = x_{ij}$ for some record i, the predicate $X_j \neq x_j$ is equivalent to the predicate $X_j = x_{ij}$ for some value of i. $\underline{\mathcal{C}}$ can thus be viewed as a *monotone* circuit over the $m + mN$ attribute equality predicates $X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m$, and $X_j = x_{ij}$ for each i and j. It follows that a database D with N records and m columns can be transformed into a database $\underline{\mathsf{D}}'$ with N records and $m + mN$ columns such that obfuscating D over the circuit $\mathcal{C}$ is equivalent to obfuscating $\underline{\mathsf{D}}$ over the monotone circuit $\underline{\mathcal{C}}$.

5.6 Alternative privacy policies

In general, a privacy policy can be any computable, possibly randomized, joint function of the database and the query. Clearly, it may be useful to consider generalizations of our privacy policies in several directions.

First, we discuss alternatives to Definition 2 that may be used to model the requirement that accessing individual records should be easy, but mass harvesting of records should be hard. To motivate this discussion, let us consider a small database with, say, 10 or 20 records. For such a database, the group-exponential property is meaningless. Even if *all* records match the adversary's query, he can easily try all 2^{10} or 2^{20} possibilities for the random bits r_{ik} because database accesses are noninteractive.

This does not in any way violate our definition of privacy. Exactly the same attack is possible against the ideal functionality, therefore, the simulation argument goes through, showing that the obfuscated database leaks no more information than the ideal functionality. It is thus natural to seek an alternative privacy definition that will make the above attack infeasible when N is small (especially when $N < k$, the security parameter).

Our construction can be easily modified to support a wide variety of (monotonically decreasing) functions capturing the dependence between the probability of the ideal functionality returning the protected attributes and the number of records matching the query. For example, the following *threshold* ideal functionality can be implemented using a threshold *(n-t)-out-of-n* secret sharing scheme [227].

- $C_{\mathsf{D}}(x, i, j)$ is 1 if $x = x_{ij}$ and 0 otherwise, where $1 \leq i \leq N, 1 \leq j \leq m$.
- $R_{\mathsf{D}}(\mathcal{P}) = \bigcup_{1 \leq i \leq N} \{\langle i, \gamma_i \rangle\}$, where

$$\gamma_i = \begin{cases} \overrightarrow{y_i} \quad 2^{-|\mathsf{D}_{[\mathcal{P}]}|} & \text{if } \mathcal{P}(\rho_i) \text{ and } |\mathsf{D}_{[\mathcal{P}]}| \leq t \\ \perp \quad \text{if } \mathcal{P}(\rho_i) \text{ and } |\mathsf{D}_{[\mathcal{P}]}| > t \\ \perp \quad \text{if } \neg\mathcal{P}(\rho_i) \end{cases}$$

The adversary can evaluate the query if there are at most t matching records, but learns nothing otherwise. The details of the construction are deferred to the full version.

We may also consider which query language should be permitted by the privacy policy. We demonstrated how to obfuscate databases in accordance with any privacy policy that permits evaluation of some predicate consisting of equality tests over database attributes. Such queries can be considered a generalization of "partial match" searches [219], which is a common query model in the database literature. Also, our algorithms can be easily modified to support policies that forbid some attributes from having $*$ as a legal value, *i.e.*, policies that require the retriever to supply the correct value for one or more designated attributes before he can extract anything from the obfuscated database.

It is worth asking if we can allow predicates over primitives looser than exact attribute equality (*e.g.*, proximity queries of [78] are an interesting class). We present strong evidence that this is impossible with our privacy definitions. In fact, even using ideal functionalities (IF) that are *more* restrictive than the one we have used does not seem to help. Recall that the IF considered in section 5.4 consists of two functions: C_{D} (it tells the retriever whether his guess of a particular query attribute value is correct) and R_{D} (it evaluates the query with the inverse-exponential probability). We will call this IF the **permissive IF**.

We define two more IFs. The **strict IF** is like the permissive IF except that it doesn't have the function C. The **semi-permissive IF** falls in between the two. It, too, doesn't have the function C, but its retrieval function R leaks slightly more information. Instead of the same symbol $\perp$, function R of the semi-permissive IF gives *different* responses depending on whether it failed to evaluate the query because it matches no records (no-matches) or because it matches too many records, and the probability came out to the retriever's disadvantage (too-many-matches).

Define $R_{\mathsf{D}}(\mathcal{P})$ as $\bigcup_{1 \leq i \leq N} R^*(\mathcal{P}, i)$, where R^* is as follows:

- If $\neg\mathcal{P}(\rho_i)$, then $R^*(\mathcal{P}, i) = \phi$.
- If $\mathcal{P}(\rho_i)$, then $R^*(\mathcal{P}, i) = \overrightarrow{y_i}$ with probability $2^{-|\mathsf{D}_{[\mathcal{P}]}|}$ and $\perp$ with probability $1 - 2^{-|\mathsf{D}_{[\mathcal{P}]}|}$.

Observe that if, for any privacy policy allowing single-attribute equality tests, *i.e.*, if all queries of the form $X_j = x_j$ are permitted, then the semi-permissive IF can simulate the permissive IF. Of course, the permissive IF can always simulate the semi-permissive IF.

We say that a privacy policy *leaks* query attributes if all x_{ij} can be computed (with overwhelming probability) simply by accessing the corresponding ideal functionality $\mathcal{I}_{\mathsf{D}}$, *i.e.*, there exists a probabilistic poly-time oracle algorithm A s.t., for any database D, $\mathbf{P}(A^{\mathcal{I}_{\mathsf{D}},\mathcal{O}}(i,j) = x_{ij}) \geq 1 - \nu(k)$. Note that the order of quantifiers has been changed: the algorithm A is now independent of the database. This captures the idea that A has no knowledge of the specific query attributes, yet successfully retrieves them with access only to the ideal functionality. Such a policy, even if securely realized, provides no meaningful privacy.

We have the following results (proofs omitted):

- If $\mathcal{X} = \{1, 2, \ldots M\}$ and queries consisting of conjunctions over inequalities are allowed, then the semi-permissive IF leaks query attributes. Each of the x_{ij} can be separately computed by binary search using queries of the form $X_j \geq x_{low} \wedge X_j \leq x_{high}$.

- If arbitrary (pptime-computable) queries are allowed, then even the strict IF leaks query attributes.

Note that a policy does not have to leak all query attributes to be intuitively useless or vacuous. For instance, a policy which allows the retriever to evaluate conjunctions of inequalities on the first $m-1$ query attributes, and allows no queries involving the last attribute, is vacuous for the semi-permissive IF. Therefore, we give a stronger criterion for vacuousness, which formalizes the notion that "all information contained in the IF can be extracted without knowing anything about the query attributes". Note that the definition below applies to arbitrary privacy policies, for it makes no reference to query or data attributes.

Definition 3 (VACUOUS PRIVACY POLICY) *We say that an ideal functionality* $\mathcal{I}_{\mathsf{D}}$ *is* vacuous *if there exists an efficient* extractor *Ext such that for any pptime algorithm A there exists a simulator S so that for any database* D*:*

$$|\mathbf{P}(A^{\mathcal{I}_{\mathsf{D}}}(1^k) = 1) - \mathbf{P}(S(Ext^{\mathcal{I}_{\mathsf{D}}}(1^k))) = 1)| = \nu(k)$$

In other words, we first extract all useful information from $\mathcal{I}_{\mathsf{D}}$ without any specific knowledge of the database, throw away $\mathcal{I}_{\mathsf{D}}$, and use the extracted information to simulate $\mathcal{I}_{\mathsf{D}}$ against an arbitrary adversary. As a special case, if *Ext* can recover the entire database D from $\mathcal{I}_{\mathsf{D}}$, then the functionality can be simulated, because the privacy policy is required to be computable and the simulator is not required to be computationally bounded (if we consider only privacy policies which are computable in probabilistic polynomial time, then we can define vacuousness with a PPT simulator as well). At the other extreme, the ideal functionality that permits no queries is also simulatable: *Ext* simply outputs nothing. The reader may verify that the IF in the all-but-one-query-attribute example above is also vacuous.

Theorem 4 *The strict ideal functionality that permits arbitrary queries is vacuous.*

Finally, we consider what happens if we use the strict IF but don't increase the power of the query language. We conjecture the existence of very simple languages, including a language that contains only conjunctions of equality tests on attributes, which are *unrealizable* even for *single-record* databases in the sense that there is no efficient obfuscation algorithm that would make the database indistinguishable from the corresponding IF. This

can be seen as justification for the choice of the permissive, rather than strict IF for our constructions.

Conjecture 1 *The strict IF for the following query language cannot be realized even for single-record databases:* $\bigwedge_{i=1}^{2k}(X_{2i-1} = x_{2i-1} \vee X_{2i} = x_{2i})$ *where* $\forall i\ x_i \in \{0,1\}$.

Note that the only constraint on the database is that its size should be polynomial in the security parameter k, and therefore we are allowed to have $2k$ query attributes.

We expect that a proof of this conjecture will also yield a proof of the following conjecture:

Conjecture 2 *The strict IF for a query language consisting of conjunction of equality tests on k query attributes is unrealizable even for single-record databases.*

These conjectures are interesting from another perspective. They can be interpreted as statements about the impossibility of circuit obfuscation in the random oracle model. They also motivate the question: given a query language, it is possible to achieve the group-exponential property with the strict IF provided there exists an obfuscation algorithm for this query language on a single record? In other words, given a class of predicates over single records and an efficient obfuscator for the corresponding circuit class, does there exist an obfuscator for the entire database that realizes the group-exponential ideal functionality for that query language? We defer a discussion of this question.

5.7 Summary

We introduced a new concept of database privacy in the offline or non-interactive setting, which is based on permitted queries rather than secrecy of individual records, and realized it using provably secure obfuscation techniques. The lesson from our result is that surprisingly strong privacy protection may be possible even in the non-interactive setting if the desired functionality of the database can be precisely described.

While our constructions are secure in the "virtual black-box" model for obfuscation, the blow-up in the size of the obfuscated database may render our techniques impractical for large databases. Our query language permits any predicate over equalities on database attributes, but other query languages may also be realizable. We define group privacy in terms of a particular ideal functionality, but there may be other functionalities that better capture intuitive security against "mass harvesting" queries. In general, investigating which ideal functionalities for database privacy can be securely realized is an important topic of future research.

Chapter 6

Robust De-anonymization of Large Sparse Datasets

6.1 Introduction

Datasets containing *micro-data*, that is, information about specific individuals, are increasingly becoming public in response to "open government" laws and to support data mining research. Some datasets include legally protected information such as health histories; others contain individual preferences and transactions, which many people may view as private or sensitive.

Privacy risks of publishing micro-data are well-known. Even if identifiers such as names and Social Security numbers have been removed, the adversary can use background knowledge and cross-correlation with other databases to re-identify individual data records. Famous attacks include de-anonymization of a Massachusetts hospital discharge database by joining it with a public voter database [239] and privacy breaches caused by (ostensibly

anonymized) AOL search data [129].

Micro-data is characterized by high dimensionality and sparsity. Each record contains many attributes (*i.e.*, columns in a database schema), which can be viewed as dimensions. Sparsity means that a pair of random records are located far apart in the multi-dimensional space defined by the attributes. This sparsity is empirically well-established [46, 16, 159] and related to the "fat tail" phenomenon: individual transaction and preference records tend to include statistically rare attributes.

Our contributions. Our first contribution is a formal model for privacy breaches in anonymized micro-data (Section 6.3). We present two definitions, one based on the probability of successful de-anonymization, the other on the amount of information recovered about the target. Unlike previous work [239], we do not assume *a priori* that the adversary's knowledge is limited to a fixed set of "quasi-identifier" attributes. Our model thus encompasses a much broader class of de-anonymization attacks than simple cross-database correlation.

Our second contribution is a very general class of de-anonymization algorithms, demonstrating the fundamental limits of privacy in public micro-data (section 6.4). Under very mild assumptions about the distribution from which the records are drawn, the adversary with a small amount of background knowledge about an individual can use it to identify, with high probability, this individual's record in the anonymized dataset and to learn all anonymously released information about him or her, including sensitive attributes. For *sparse* datasets, such as most real-world datasets of individual transactions, preferences, and recommendations, very little background knowledge is needed

(as few as 5-10 attributes in our case study). Our de-anonymization algorithm is *robust* to the imprecision of the adversary's background knowledge and to perturbation that may have been applied to the data prior to release. It works even if only a *subset* of the original dataset has been published.

Our third contribution is a practical analysis of the Netflix Prize dataset, containing anonymized movie ratings of 500,000 Netflix subscribers (section 6.5). Netflix—the world's largest online DVD rental service—published this dataset to support the Netflix Prize data mining contest. We demonstrate that an adversary who knows a little bit about some subscriber can easily identify her record if it is present in the dataset, or, at the very least, identify a small set of records which include the subscriber's record. The adversary's background knowledge need not be precise, *e.g.*, the dates may only be known to the adversary with a 14-day error, the ratings may be known only approximately, and some of the ratings and dates may even be completely wrong. Because our algorithm is robust, if it uniquely identifies a record in the published dataset, with high probability this identification is not a false positive.

6.2 Related work

Unlike statistical databases [6, 13, 35], micro-data include actual records of individuals even after anonymization. A popular approach to micro-data privacy is k-anonymity [241, 62]. The data publisher decides in advance which of the attributes may be available to the adversary (these are called "quasi-identifiers"), and which are the sensitive attributes to be protected. k-anonymization ensures that each quasi-identifier tuple occurs in at least k records in the anonymized database. This does not guarantee any privacy, be-

cause the values of sensitive attributes associated with a given quasi-identifier may not be sufficiently diverse [172, 173] or the adversary may know more than just the quasi-identifiers [172]. Furthermore, k-anonymization completely fails on high-dimensional datasets [10], such as the Netflix Prize dataset and most real-world datasets of individual recommendations and purchases.

The de-anonymization algorithm presented in this paper does not assume that the attributes are divided *a priori* into quasi-identifiers and sensitive attributes. Examples include anonymized transaction records (if the adversary knows a few of the individual's purchases, can he learn *all* of her purchases?), recommendations and ratings (if the adversary knows a few movies that the individual watched, can he learn *all* movies she watched?), Web browsing and search histories, and so on. In such datasets, it is hard to tell in advance which attributes might be available to the adversary; the adversary's background knowledge may even vary from individual to individual. Unlike [239, 174, 102], our algorithm is *robust*. It works even if the published records have been perturbed, if only a subset of the original dataset has been published, and if there are mistakes in the adversary's background knowledge.

Our definition of privacy breach is somewhat similar to that of Chawla *et al.* [56]. We discuss the differences in section 6.3. There is theoretical evidence that for any (sanitized) database with meaningful utility, there is *always* some auxiliary or background information that results in a privacy breach [81]. In this paper, we aim to quantify the amount of auxiliary information required and its relationship to the percentage of records which would experience a significant privacy loss.

We are aware of only one previous paper that considered privacy of movie ratings. In collaboration with the MovieLens recommendation service,

Frankowski *et al.* correlated public mentions of movies in the MovieLens discussion forum with the users' movie rating histories in the internal MovieLens dataset [102]. The algorithm uses the entire public record as the background knowledge (29 ratings per user, on average), and is not robust if this knowledge is imprecise, *e.g.*, if the user publicly mentioned movies which he did not rate.

While our algorithm follows the same basic scoring paradigm as [102], our scoring function is more complex and our selection criterion is nontrivial and an important innovation in its own right. Furthermore, our case study is based solely on public data and does *not* involve cross-correlating internal Netflix datasets (to which we do not have access) with public forums. It requires much less background knowledge (2-8 ratings per user), which need not be precise. Furthermore, our analysis has privacy implications for 500,000 Netflix subscribers whose records have been published; by contrast, the largest public MovieLens datasets contains only 6,000 records.

6.3 Model

Database. Define database $\mathcal{D}$ to be an $N \times M$ matrix where each row is a record associated with some individual, and the columns are attributes. We are interested in databases containing individual preferences or transactions. The number of columns thus reflects the total number of items in the space we are considering, ranging from a few thousand for movies to millions for (say) the `amazon.com` catalog.

Each attribute (column) can be thought of as a dimension, and each individual record as a point in the multidimensional attribute space. To keep

our analysis general, we will not fix the space X from which attributes are drawn. They may be boolean (*e.g.*, has this book been rated?), integer (*e.g.*, the book's rating on a 1-10 scale), date, or a tuple such as a (rating, date) pair.

A typical reason to publish anonymized micro-data is "collaborative filtering," *i.e.*, predicting a consumer's future choices from his past behavior using the knowledge of what similar consumers did. Technically, the goal is to predict the value of some attributes using a combination of other attributes. This is used in shopping recommender systems, aggressive caching in Web browsers, and other applications [245].

Sparsity and similarity. Preference databases with thousands of attributes are necessarily *sparse*, *i.e.*, each individual record contains values only for a small fraction of attributes. For example, the shopping history of even the most profligate Amazon shopper contains only a tiny fraction of all available items. We call these attributes *non-null*; the set of non-null attributes is the *support* of a record (denoted $\mathsf{supp}(r)$). Null attributes are denoted $\perp$. The support of a column is defined analogously. Even though points corresponding to database records are very sparse in the attribute space, each record may have dozens or hundreds of non-null attributes, making the database truly high-dimensional.

The distribution of per-attribute support sizes is typically heavy- or *long-tailed*, roughly following the power law [46, 16]. This means that although the supports of the columns corresponding to "unpopular" items are small, these items are so numerous that they make up the bulk of the non-null entries in the database. Thus, any attempt to approximate the database by projecting

it down to the most common columns is bound to failure.[1]

Unlike "quasi-identifiers" [241, 62], there are no attributes that can be used directly for de-anonymization. In a large database, for any except the rarest attributes, there are hundreds of records with the same value of this attribute. Therefore, it is *not* a quasi-identifier. At the same time, knowledge that a particular individual has a certain attribute value does reveal *some* information, since attribute values and even the mere fact that a given attribute is non-null vary from record to record.

The similarity measure Sim is a function that maps a pair of attributes (or more generally, a pair of records) to the interval $[0, 1]$. It captures the intuitive notion of two values being "similar." Typically, Sim on attributes will behave like an indicator function. For example, in our analysis of the Netflix Prize dataset, Sim outputs 1 on a pair of movies rated by different subscribers if and only if both the ratings and the dates are within a certain threshold of each other; it outputs 0 otherwise.

We define Sim over two records r_1, r_2 by generalizing the similarity measure from individual attributes to vectors of attributes:

$$\mathsf{Sim}(r_1, r_2) = \frac{\sum \mathsf{Sim}(r_{1i}, r_{2i})}{|\mathsf{supp}(r_1) \cup \mathsf{supp}(r_2)|}$$

Definition 4 (Sparsity) *A database $\mathcal{D}$ is (ϵ, δ)-sparse w.r.t. the similarity measure Sim if*

$$\Pr_r[\mathsf{Sim}(r, r') > \epsilon \ \forall r' \neq r] \leq \delta$$

As a real-world example, in fig. 6.1 we show that the Netflix Prize dataset is overwhelmingly sparse. For the vast majority of records, there

[1]The same effect causes k-anonymization to fail on high-dimensional databases [10].

isn't a *single* record with similarity score over 0.5 in the entire 500,000-record dataset, even if we consider only the sets of movies rated without taking into account numerical ratings or dates.

Sanitization and sampling. Database sanitization methods include generalization and suppression [240, 62], as well as perturbation. The data publisher may only release a (possibly non-uniform) sample of the database. Our algorithm is designed to work against data that have been both anonymized and sanitized.

If the database is published for collaborative filtering or similar data mining purposes (as in the case of the Netflix Prize dataset), the "error" introduced by sanitization *cannot* be large, otherwise data utility will be lost. We make this precise in our analysis. Our definition of privacy breach allows the adversary to identify not just his target record, but *any* record as long as it is sufficiently similar (via Sim) to the target and can thus be used to determine its attributes with high probability.

From the viewpoint of our de-anonymization algorithm, there is no difference between the perturbation of the published records and the imprecision of the adversary's knowledge about his target. In either case, there is a small discrepancy between the attribute value(s) in the anonymous record and the same value(s) as known to the adversary. In the rest of the paper, we treat perturbation simply as imprecision of the adversary's knowledge. The algorithm is designed to be robust to the latter.

Adversary model. We sample record r randomly from database $\mathcal{D}$ and give *auxiliary information* or background knowledge related to r to the adversary. It is restricted to a subset of (possibly imprecise, perturbed, or simply incor-

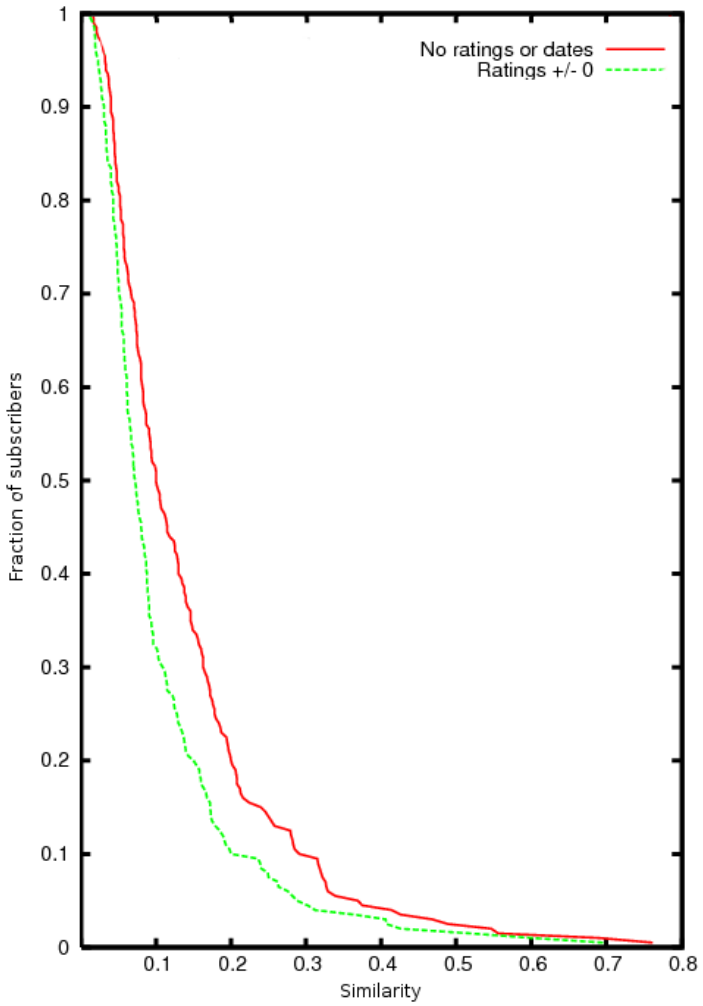

Figure 6.1: X-axis (x) is the similarity to the "neighbor" with the highest similarity score; Y-axis is the fraction of subscribers whose nearest-neighbor similarity is at least x.

rect) values of r's attributes, modeled as an arbitrary probabilistic function Aux: $X^M \to X^M$. The attributes given to the adversary may be chosen uniformly from the support of r, or according to some other rule.[2] Given this auxiliary information and an anonymized sample $\hat{\mathcal{D}}$ of $\mathcal{D}$, the adversary's goal is to reconstruct attribute values of the entire record r. Note that there is no artificial distinction between quasi-identifiers and sensitive attributes.

If the published records are sanitized by adding random noise Z_S, and the noise used in generating Aux is Z_A, then the adversary's task is equivalent to the scenario where the data is not perturbed but noise $Z_S + Z_A$ is used in generating Aux. This makes perturbation equivalent to imprecision of Aux.

Privacy breach: formal definitions. What does it mean to de-anonymize a record r? The naive answer is to find the "right" anonymized record in the public sample $\hat{\mathcal{D}}$. This is hard to capture formally, however, because it requires assumptions about the data publishing process (*e.g.*, what if $\hat{\mathcal{D}}$ contains two copies of every original record?). Fundamentally, the adversary's objective is is to learn as much as he can about r's attributes that he doesn't already know. We give two different (but related) formal definitions, because there are two distinct scenarios for privacy breaches in large databases.

The first scenario is automated large-scale de-anonymization. For every record r about which he has some information, the adversary must produce a single "prediction" for all attributes of r. An example is the attack that inspired k-anonymity [239]: taking the demographic data from a voter database as auxiliary information, the adversary joins it with the anonymized hospital

[2]For example, in the Netflix Prize case study we also pick uniformly from among the attributes whose supports are below a certain threshold, *e.g.*, movies that are outside the most popular 100 or 500 movies.

discharge database and uses the resulting combination to determine the values of medical attributes for each person who appears in both databases.

Definition 5 *A database $\mathcal{D}$ can be (θ, ω)-deanonymized w.r.t. auxiliary information Aux if there exists an algorithm A which, on inputs $\mathcal{D}$ and $\mathsf{Aux}(r)$ where $r \leftarrow \mathcal{D}$ outputs r' such that*

$$\Pr[\mathsf{Sim}(r, r') \geq \theta] \geq \omega$$

Definition 5 can be interpreted as an *amplification of background knowledge*: the adversary starts with $\mathsf{aux} = \mathsf{Aux}(r)$ which is close to r on a small subset of attributes, and uses this to compute r' which is close to r on the entire set of attributes. This captures the **adversary's ability to gain information about his target record**. As long he finds *some* record which is guaranteed to be very similar to the target record, *i.e.*, contains the same or similar attribute values, privacy breach has occurred.

If operating on a sample $\hat{\mathcal{D}}$, the de-anonymization algorithm must also detect whether the target record is part of the sample, or has not been released at all. In the following, the probability is taken over the randomness of the sampling of r from $\hat{\mathcal{D}}$, Aux and A itself.

Definition 6 (De-anonymization) *An arbitrary subset $\hat{\mathcal{D}}$ of a database $\mathcal{D}$ can be (θ, ω)-deanonymized w.r.t. auxiliary information Aux if there exists an algorithm A which, on inputs $\hat{\mathcal{D}}$ and $\mathsf{Aux}(r)$ where $r \leftarrow \mathcal{D}$*

- *If $r \in \hat{\mathcal{D}}$, outputs r' s.t.* $\Pr[\mathsf{Sim}(r, r') \geq \theta] \geq \omega$
- *if $r \notin \hat{\mathcal{D}}$, outputs $\perp$ with probability at least ω*

The same error threshold $(1 - \omega)$ is used for both false positives and false negatives because the parameters of the algorithm can be adjusted so that both rates are equal; this is the "equal error rate."

In the second privacy breach scenario, the adversary produces a set or "lineup" of candidate records that include his target record r, either because there is not enough auxiliary information to identify r in the lineup or because he expects to perform additional analysis to complete de-anonymization. This is similar to communication anonymity in mix networks [226].

The *number* of candidate records is not a good metric, because some of the records may be much likelier candidates than others. Instead, we consider the probability distribution over the candidate records, and use as the metric the conditional *entropy* of r given aux. In the absence of an "oracle" to identify the target record r in the lineup, the entropy of the distribution itself can be used as a metric [226, 76]. If the adversary has such an "oracle" (this is a technical device used to measure the adversary's success; in the real world, the adversary may not have an oracle telling him whether de-anonymization succeeded), then privacy breach can be quantified as follows: *how many bits of additional information does the adversary need in order to output a record which is similar to his target record?*

Thus, suppose that after executing the de-anonymization algorithm, the adversary outputs records $r'_1, \ldots r'_k$ and the corresponding probabilities $p_1, \ldots p_k$. The latter can be viewed as an *entropy encoding* of the candidate records. According to Shannon's source coding theorem, the optimal code length for record r'_i is $(-\log p_i)$. We denote by $H_S(\Pi, x)$ this Shannon entropy of a record x w.r.t. a probability distribution Π. In the following, the expectation is taken over the coin tosses of A, the sampling of r and Aux.

Definition 7 (Entropic de-anonymization) *A database $\mathcal{D}$ can be (θ, H)-deanonymized w.r.t. auxiliary information* Aux *if there exists an algorithm A which, on inputs $\mathcal{D}$ and $\mathsf{Aux}(r)$ where $r \leftarrow \mathcal{D}$ outputs a set of candidate records $\mathcal{D}'$ and probability distribution Π such that*

$$E[\mathsf{min}_{r' \in \mathcal{D}', \mathsf{Sim}(r,r') \geq \theta} H_S(\Pi, r')] \leq H$$

This definition measures the minimum Shannon entropy of the candidate set of records which are similar to the target record. As we will show, in sparse databases this set is likely to contain a single record, thus taking the minimum is but a syntactic requirement.

When the minimum is taken over an empty set, we define it to be $H_0 = \log_2 N$, the *a priori* entropy of the target record. This models outputting a random record from the entire database when the adversary cannot compute a lineup of plausible candidates. Formally, the adversary's algorithm A can be converted into an algorithm A', which outputs the mean of two distributions: one is the output of A, the other is the uniform distribution over $\mathcal{D}$. Observe that for A', the minimum is always taken over a non-empty set, and the expectation for A' differs from that for A by at most 1 bit.

Chawla *et al.* [56] give a definition of privacy breach via *isolation* which is similar to ours, but requires a metric on attributes, whereas our general similarity measure does not naturally lead to a metric (there is no feasible way to derive a distance function from it that satisfies the triangle inequality). This appears to be essential for achieving robustness to completely erroneous attributes in the adversary's auxiliary information.

6.4 De-anonymization algorithm

We start by describing an algorithm template or meta-algorithm. The inputs are a sample $\hat{\mathcal{D}}$ of database $\mathcal{D}$ and auxiliary information $\mathsf{aux} = Aux(r), r \leftarrow \mathcal{D}$. The output is either a record $r' \in \hat{\mathcal{D}}$, or a set of candidate records and a probability distribution over those records (following Definitions 6 and 7, respectively).

The three main components of the algorithm are the scoring function, matching criterion, and record selection. The **scoring function** Score assigns a numerical score to each record in $\hat{\mathcal{D}}$ based on how well it matches the adversary's auxiliary information Aux. The **matching criterion** is the algorithm applied by the adversary to the set of scores to determine if there is a match. Finally, **record selection** selects one "best-guess" record or a probability distribution, if needed.

1. Compute $\mathsf{Score}(\mathsf{aux}, r')$ for each $r' \in \hat{\mathcal{D}}$.

2. Apply the matching criterion to the resulting set of scores and compute the matching set; if the matching set is empty, output $\perp$ and exit.

3. If a "best guess" is required (de-anonymization according to Defs. 5 and 6), output $r' \in \hat{\mathcal{D}}$ with the highest score. If a probability distribution over candidate records is required (de-anonymization according to Def. 7), compute and output some non-decreasing distribution based on the scores.

Algorithm Scoreboard. The following simple instantiation of the above template is sufficiently tractable to be formally analyzed in the rest of this section.

- $\mathsf{Score}(\mathsf{aux}, r') = \mathsf{min}_{i \in \mathsf{supp}(\mathsf{aux})} \mathsf{Sim}(\mathsf{aux}_i, r'_i)$, *i.e.*, the score of a candidate record is determined by the least similar attribute between it and the adversary's auxiliary information.

- The matching set $\mathcal{D}' = \{r' \in \hat{\mathcal{D}} : \mathsf{Score}(\mathsf{aux}, r') > \alpha\}$ for some fixed constant α. The matching criterion is that $\mathcal{D}'$ be nonempty.

- Probability distribution is uniform on $\mathcal{D}'$.

Algorithm Scoreboard-RH. Algorithm Scoreboard is not sufficiently robust for some applications; in particular, it fails if any of the attributes in the adversary's auxiliary information are completely incorrect.

The following algorithm incorporates several heuristics which have proved useful in practical analysis (see section 6.5). First, the scoring function gives higher weight to statistically rare attributes. Intuitively, if the auxiliary information tells the adversary that his target has a certain rare attribute, this helps de-anonymization much more than the knowledge of a common attribute (*e.g.*, it is more useful to know that the target has purchased "The Dedalus Book of French Horror" than the fact that she purchased a Harry Potter book).

Second, to improve robustness, the matching criterion requires that the top score be significantly above the second-best score. This measures how much the identified record "stands out" from other candidate records.

- $\mathsf{Score}(\mathsf{aux}, r') = \sum_{i \in \mathsf{supp}(\mathsf{aux})} \mathsf{wt}(i) \mathsf{Sim}(\mathsf{aux}_i, r'_i)$ where $\mathsf{wt}(i) = \frac{1}{\log |\mathsf{supp}(i)|}$. [3]

- If a "best guess" is required, compute $\mathsf{max} = \mathsf{max}(S), \mathsf{max}_2 = \mathsf{max}_2(S)$ and $\sigma = \sigma(S)$ where $S = \{\mathsf{Score}(\mathsf{aux}, r') : r' \in \hat{\mathcal{D}}\}$, *i.e.*, the highest

[3] Without loss of generality, we assume $\forall i \ |\mathsf{supp}(i)| > 0$.

and second-highest scores and the standard deviation of the scores. If $\frac{\mathsf{max}-\mathsf{max}_2}{\sigma} < \phi$, where ϕ is a fixed parameter called the *eccentricity*, then there is no match; otherwise, the matching set consists of the record with the highest score.[4]

- If entropic de-anonymization is required, output distribution $\Pi(r') = c \cdot e^{\frac{\mathsf{Score}(\mathsf{aux}, r')}{\sigma}}$ for each r', where c is a constant that makes the distribution sum up to 1. This weighs each matching record in inverse proportion to the likelihood that the match in question is a statistical fluke.

Note that there are two ways in which this algorithm can fail to find the correct record. First, an incorrect record may be assigned the highest score. Second, the correct record may not have a score which is significantly higher than the second-highest score.

6.4.1 Analysis: general case

We now quantify the amount of auxiliary information needed to de-anonymize an arbitrary dataset using Algorithm Scoreboard. The smaller the required information (*i.e.*, the fewer attribute values the adversary needs to know about his target), the easier the attack.

We start with the worst-case analysis and calculate how much auxiliary information is needed without any assumptions about the distribution from which the data is drawn. In section 6.4.2, we will show that much less auxiliary information is needed to de-anonymize records drawn from *sparse* distributions (real-world transaction and recommendation datasets are all sparse).

[4]Increasing ϕ increases the false negative rate, *i.e.*, the chance of erroneously dismissing a correct match, and decreases the false positive rate; ϕ may be chosen so that the two rates are equal.

Let aux be the auxiliary information about some record r; aux consists of m (non-null) attribute values, which are close to the corresponding values of attributes in r, that is, $|\mathsf{aux}| = m$ and $\mathsf{Sim}(\mathsf{aux}_i, r_i) \geq 1 - \epsilon \ \forall i \in \mathsf{supp}(\mathsf{aux})$, where aux_i (respectively, r_i) is the ith attribute of aux (respectively, r).

Theorem 5 *Let $0 < \epsilon, \delta < 1$ and let $\mathcal{D}$ be the database. Let Aux be such that $\mathsf{aux} = \mathsf{Aux}(r)$ consists of at least $m \geq \frac{\log N - \log \epsilon}{-\log(1-\delta)}$ randomly selected attribute values of the target record r, where $\forall i \in \mathsf{supp}(\mathsf{aux})$ $\mathsf{Sim}(\mathsf{aux}_i, r_i) \geq 1-\epsilon$. Then $\mathcal{D}$ can be $(1 - \epsilon - \delta, 1 - \epsilon)$-deanonymized w.r.t. Aux.*

Proof. Use Algorithm $\mathsf{Scoreboard}$ with $\alpha = 1 - \epsilon$ to compute the set of all records in $\hat{\mathcal{D}}$ that match aux, then output a record r' at random from the matching set. It is sufficient to prove that this randomly chosen r' must be very similar to the target record r. (This satisfies our definition of a privacy breach because it gives the adversary almost everything he may want to learn about r.)

Record r' is a *false match* if $\mathsf{Sim}(r, r') \leq 1-\epsilon-\delta$ (*i.e.*, the likelihood that r' is similar to the target r is below the threshold). We first show that, with high probability, there are no false matches in the matching set.

Lemma 1 *If r' is a false match, then $\Pr_{i \in \mathsf{supp}(r)}[\mathsf{Sim}(r_i, r'_i) \geq 1 - \epsilon] < 1 - \delta$*

Lemma 1 holds, because the contrary implies $\mathsf{Sim}(r, r') \geq (1-\epsilon)(1-\delta) \geq (1 - \epsilon - \delta)$, contradicting the assumption that r' is a false match. Therefore, the probability that the false match r' belongs to the matching set is at most $(1 - \delta)^m$. By a union bound, the probability that the matching set contains even a single false match is at most $N(1 - \delta)^m$. If $m = \frac{\log \frac{N}{\epsilon}}{\log \frac{1}{1-\delta}}$, then the

probability that the matching set contains any false matches is no more than ϵ.

Therefore, with probability $1 - \epsilon$, there are no false matches. Thus for every record r' in the matching set, $\mathsf{Sim}(r, r') \geq 1 - \epsilon - \delta$, *i.e.*, any r' must be similar to the true record r. To complete the proof, observe that the matching set contains at least one record, r itself.

When δ is small, $m = \frac{\log N - \log \epsilon}{\delta}$. This depends logarithmically on ϵ and linearly on δ: the chance that the algorithm fails completely is very small even if attribute-wise accuracy is not very high. Also note that the matching set need not be small. Even if the algorithm returns many records, with high probability they are *all* similar to the target record r, and thus any one of them can be used to learn the unknown attributes of r.

6.4.2 Analysis: sparse datasets

Most real-world datasets containing individual transactions, preferences, and so on are *sparse*. Sparsity increases the probability that de-anonymization succeeds, decreases the amount of auxiliary information needed, and improves robustness to both perturbation in the data and mistakes in the auxiliary information.

Our assumptions about data sparsity are very mild. We only assume $(1 - \epsilon - \delta, \ldots)$ sparsity, *i.e.*, we assume that the average record does not have *extremely* similar peers in the dataset (real-world records tend not to have even *approximately* similar peers—see fig. 6.1).

Theorem 6 *Let ϵ, δ, and aux be as in Theorem 5. If the database $\mathcal{D}$ is $(1 - \epsilon - \delta, \epsilon)$-sparse, then $\mathcal{D}$ can be $(1, 1 - \epsilon)$-deanonymized.* □

The proof is essentially the same as for Theorem 5, but in this case *any* $r' \neq r$ from the matching set must be a false match. Because with probability $1-\epsilon$, Scoreboard outputs no false matches, the matching set consists of exactly one record: the true target record r.

De-anonymization in the sense of Definition 7 requires even less auxiliary information. Recall that in this kind of privacy breach, the adversary outputs a "lineup" of k suspect records, one of which is the true record. This k-deanonymization is equivalent to $(1, \frac{1}{k})$-deanonymization in our framework.

Theorem 7 *Let $\mathcal{D}$ be $(1-\epsilon-\delta, \epsilon)$-sparse and* aux *be as in Theorem 5 with $m = \frac{\log \frac{N}{k-1}}{\log \frac{1}{1-\delta}}$. Then*

- *$\mathcal{D}$ can be $(1, \frac{1}{k})$-deanonymized.*
- *$\mathcal{D}$ can be $(1, \log k)$-deanonymized (entropically).*

By the same argument as in the proof of Theorem 5, if the adversary knows $m = \frac{\log \frac{N}{k-1}}{\log \frac{1}{1-\delta}}$ attributes, then the expected number of false matches in the matching set is at most $k-1$. Let X be the random variable representing this number. A random record from the matching set is a false match with probability of at least $\frac{1}{X}$. Since $\frac{1}{x}$ is a convex function, apply Jensen's inequality [140] to obtain $E[\frac{1}{X}] \geq \frac{1}{E(X)} \geq \frac{1}{k}$.

Similarly, if the adversary outputs the uniform distribution over the matching set, its entropy is $\log X$. Since $\log x$ is a concave function, by Jensen's inequality $E[\log X] \leq \log E(X) \leq \log k$.

Neither claim follows directly from the other. □

6.4.3 De-anonymization from a sample

We now consider the scenario in which the released database $\hat{\mathcal{D}} \subsetneq \mathcal{D}$ is a sample of the original database $\mathcal{D}$, *i.e.*, only some of the anonymized records are available to the adversary. This is the case, for example, for the Netflix Prize dataset (the subject of our case study in section 6.5), where the publicly available anonymized sample contains less than $\frac{1}{10}$ of the original data.

In this scenario, even though the original database $\mathcal{D}$ contains the adversary's target record r, this record may not appear in $\hat{\mathcal{D}}$ even in anonymized form. The adversary can still apply Scoreboard, but the matching set may be empty, in which case the adversary outputs $\perp$ (indicating that de-anonymization fails). If the matching set is not empty, he proceeds as before: picks a random record r' and learn the attributes of r on the basis of r'. We now demonstrate the equivalent of Theorem 5: de-anonymization succeeds as long as r is in the public sample; otherwise, the adversary can detect, with high probability, that r is not in the public sample.

Theorem 8 *Let ϵ, δ, $\mathcal{D}$, and aux be as in Theorem 5, and $\hat{\mathcal{D}} \subset \mathcal{D}$. Then $\hat{\mathcal{D}}$ can be $(1-\epsilon-\delta, 1-\epsilon)$-deanonymized w.r.t. aux.* □

The bound on the probability of a false match given in the proof of Theorem 5 still holds, and the adversary is guaranteed at least one match as long as his target record r is in $\hat{\mathcal{D}}$. Therefore, if $r \notin \hat{\mathcal{D}}$, the adversary outputs $\perp$ with probability at least $1-\epsilon$. If $r \in \hat{\mathcal{D}}$, then again the adversary succeeds with probability at least $1-\epsilon$.

Theorems 6 and 7 do not translate directly. For each record in the public sample $\hat{\mathcal{D}}$, there could be any number of similar records in $\mathcal{D} \setminus \hat{\mathcal{D}}$, the part of the database that is not available to the adversary.

Fortunately, if $\mathcal{D}$ is sparse, then theorems 6 and 7 still hold, and de-anonymization succeeds with a very small amount of auxiliary information. We now show that if the random sample $\hat{\mathcal{D}}$ is sparse, then the entire database $\mathcal{D}$ must also be sparse. Therefore, the adversary can simply apply the de-anonymization algorithm to the sample. If he finds the target record r, then with high probability this is not a false positive.

Theorem 9 *If database $\mathcal{D}$ is not (ϵ, δ)-sparse, then a random $\frac{1}{\lambda}$-subset $\hat{\mathcal{D}}$ is not $(\epsilon, \frac{\delta\gamma}{\lambda})$-sparse with probability at least $1 - \gamma$.* □

For each $r \in \hat{\mathcal{D}}$, the "nearest neighbor" r' of r in $\mathcal{D}$ has a probability $\frac{1}{\lambda}$ of being included in $\hat{\mathcal{D}}$. Therefore, the expected probability that the similarity with the nearest neighbor is at least $1-\epsilon$ is at least $\frac{\delta}{\lambda}$. (Here the expectation is over the set of all possible samples and the probability is over the choice of the record in $\hat{\mathcal{D}}$.) Applying Markov's inequality, the probability, taken over the choice $\hat{\mathcal{D}}$, that $\hat{\mathcal{D}}$ is sparse, *i.e.*, that the similarity with the nearest neighbor is $\frac{\delta\gamma}{\lambda}$, is no more than γ. □

The above bound is quite pessimistic. Intuitively, for any "reasonable" dataset, the sparsity of a random sample will be about the same as that of the original dataset.

Theorem 9 can be interpreted as follows. Consider the adversary who has access to a sparse sample $\hat{\mathcal{D}}$, but not the entire database $\mathcal{D}$. Theorem 9 says that either a very-low-probability event has occurred, or $\mathcal{D}$ itself is sparse. Note that it is meaningless to try to bound the probability that $\mathcal{D}$ is sparse because we do not have a probability distribution on how $\mathcal{D}$ itself is created.

Intuitively, this says that unless the sample is specially tailored, sparsity of the sample implies sparsity of the entire database. The alternative is that

the similarity between a random record in the sample and its nearest neighbor is very different from the corresponding distribution in the full database. In practice, most, if not all anonymized datasets are published to support research on data mining and collaborative filtering. Tailoring the published sample in such a way that its nearest-neighbor similarity is radically different from that of the original data would completely destroy utility of the sample for learning new collaborative filters, which are often based on the set of nearest neighbors. Therefore, in real-world anonymous data publishing scenarios—including, for example, the Netflix Prize dataset—sparsity of the sample should imply sparsity of the original dataset.

6.5 Case study: Netflix Prize dataset

On October 2, 2006, Netflix, the world's largest online DVD rental service, announced the $1-million Netflix Prize for improving their movie recommendation service [126]. To aid contestants, Netflix publicly released a dataset containing $100,480,507$ movie ratings, created by $480,189$ Netflix subscribers between December 1999 and December 2005. At the end of 2005, Netflix had approximately 4 million subscribers, so almost $\frac{1}{8}$ of them had their records published.

Among the Frequently Asked Questions about the Netflix Prize [197], there is the following question: "Is there any customer information in the dataset that should be kept private?" The answer is as follows:

> "No, all customer identifying information has been removed; all that remains are ratings and dates. This follows our privacy policy [...] Even if, for example, you knew all your own ratings and

> their dates you probably couldn't identify them reliably in the data because only a small sample was included (less than one-tenth of our complete dataset) and that data was subject to perturbation. Of course, since you know all your own ratings that really isn't a privacy problem is it?"

Removing identifying information is not sufficient for anonymity. An adversary may have auxiliary information about a subscriber's movie preferences: the titles of a few of the movies that this subscriber watched, whether she liked them or not, maybe even approximate dates when she watched them. We emphasize that even if it is hard to collect such information for a large number of subscribers, targeted de-anonymization—for example, a boss using the Netflix Prize dataset to find an employee's entire movie viewing history after a casual conversation—still presents a serious threat to privacy.

We investigate the following question: *How much does the adversary need to know about a Netflix subscriber in order to identify her record if it is present in the dataset, and thus learn her complete movie viewing history?* Formally, we study the relationship between the size of aux and $(1, \omega)$- and $(1, H)$-deanonymization.

Does privacy of Netflix ratings matter? The issue is *not* "Does the average Netflix subscriber care about the privacy of his movie viewing history?," but "Are there *any* Netflix subscribers whose privacy can be compromised by analyzing the Netflix Prize dataset?" As shown by our experiments below, it is possible to learn sensitive *non-public* information about a person from his or her movie viewing history. We assert that even if the vast majority of Netflix subscribers did not care about the privacy of their movie ratings (which is not

obvious by any means), our analysis would still indicate serious privacy issues with the Netflix Prize dataset.

Moreover, the linkage between an individual and her movie viewing history has implications for her *future* privacy. In network security, "forward secrecy" is important: even if the attacker manages to compromise a session key, this should not help him much in compromising the keys of future sessions. Similarly, one may state the "forward privacy" property: if someone's privacy is breached (*e.g.*, her anonymous online records have been linked to her real identity), future privacy breaches should not become easier. Consider a Netflix subscriber Alice whose entire movie viewing history has been revealed. Even if in the future Alice creates a brand-new virtual identity (call her Ecila), Ecila will *never* be able to disclose any non-trivial information about the movies that she had rated within Netflix because any such information can be traced back to her real identity via the Netflix Prize dataset. In general, once any piece of data has been linked to a person's *real* identity, any association between this data and a *virtual* identity breaks anonymity of the latter.

Finally, the Video Privacy Protection Act of 1988 [87] lays down strong provisions against disclosure of personally identifiable rental records of "prerecorded video cassette tapes or similar audio visual material." While the Netflix Prize dataset does not *explicitly* include personally identifiable information, the issue of whether the implicit disclosure demonstrated by our analysis runs afoul of the law or not is a legal question to be considered.

How did Netflix release and sanitize the data? Figs. 6.2 and 6.3 plot the number of ratings X against the number of subscribers in the released dataset who have at least X ratings. The tail is surprisingly thick: thousands

of subscribers have rated more than a thousand movies. Netflix claims that the subscribers in the released dataset have been "randomly chosen." Whatever the selection algorithm was, it was not uniformly random. Common sense suggests that with uniform subscriber selection, the curve would be monotonically decreasing (as most people rate very few movies or none at all), and that there would be no sharp discontinuities.

We conjecture that some fraction of subscribers with more than 20 ratings were sampled, and the points on the graph to the left of $X = 20$ are the result of some movies being deleted after sampling.

We requested the rating history as presented on the Netflix website from some of our acquaintances, and based on this data (which is effectively drawn from Netflix's *original*, non-anonymous dataset, since we know the names associated with these records), located two of them in the Netflix Prize dataset. Netflix's claim that the data was perturbed does not appear to be borne out. One of the subscribers had 1 of 306 ratings altered, and the other had 5 of 229 altered. (These are upper bounds, because the subscribers may have changed their ratings after Netflix took the 2005 snapshot that was released.) In any case, the level of noise is far too small to affect our de-anonymization algorithm, which has been specifically designed to withstand this kind of imprecision. We have no way of determining how many dates were altered and how many ratings were deleted, but we conjecture that very little perturbation has been applied.

It is important that the Netflix Prize dataset has been released to support development of better recommendation algorithms. A significant perturbation of individual attributes would have affected cross-attribute correlations and significantly decreased the dataset's utility for creating new recommenda-

tion algorithms, defeating the entire purpose of the Netflix Prize competition.

Note that the Netflix Prize dataset clearly has not been k-anonymized for any value of $k > 1$.

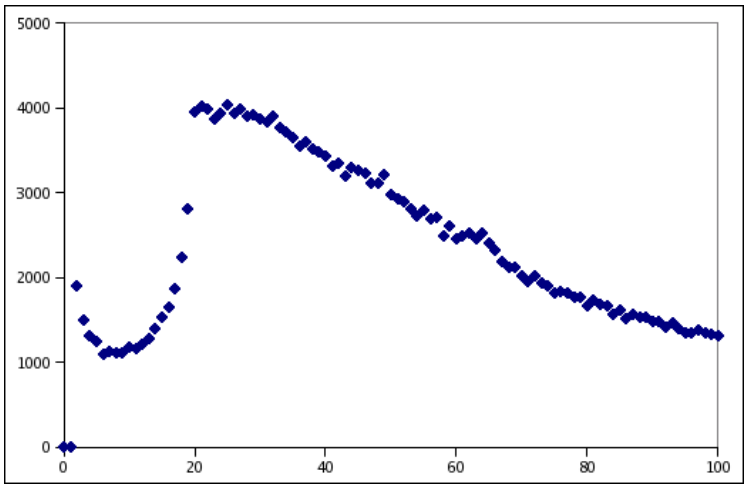

Figure 6.2: For each $k \leq 100$, the number of subscribers with k ratings in the released dataset.

De-anonymizing the Netflix Prize dataset. We apply Algorithm Scoreboard-RH from section 6.4. The similarity measure Sim on attributes is a threshold function: Sim returns 1 if and only if the two attribute values are within a certain threshold of each other. For movie ratings, which in the case of Netflix are on the 1-5 scale, we consider the thresholds of 0 (corresponding to exact match) and 1, and for the rating dates, 3 days, 14 days, or ∞. The latter means that the adversary has no information about the date when the movie was rated.

We allow some of the attribute values in the attacker's auxiliary information to be completely wrong. Thus, we say that aux of a record r consists of m movies out of m' if $|\mathsf{aux}| = m'$ and $\sum \mathsf{Sim}(\mathsf{aux}_i, r_i) \geq m$. We instantiate

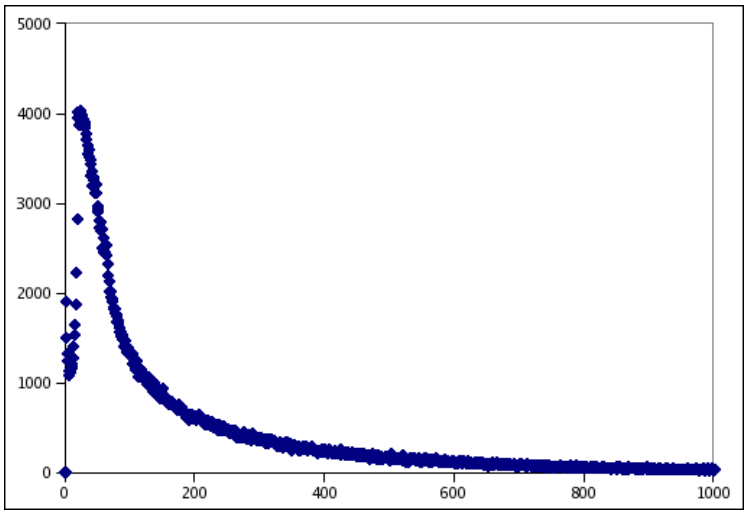

Figure 6.3: For each $k \leq 1000$, the number of subscribers with k ratings in the released dataset.

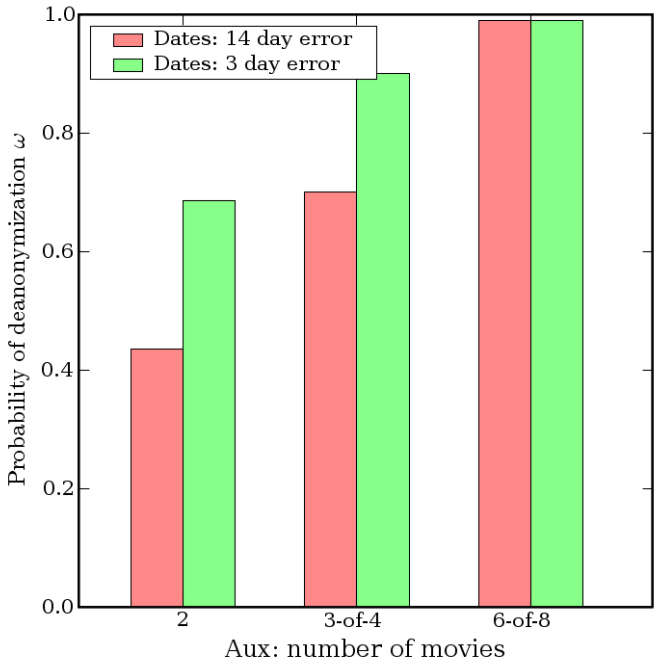

Figure 6.4: Adversary knows exact ratings and approximate dates.

the scoring function as follows:

$$\mathsf{Score}(\mathsf{aux}, r') = \sum_{i \in \mathsf{supp}(\mathsf{aux})} \mathsf{wt}(i)(e^{\frac{\rho_i - \rho_i'}{\rho_0}} + e^{\frac{d_i - d_i'}{d_0}})$$

where $\mathsf{wt}(i) = \frac{1}{\log |\mathsf{supp}(i)|}$ ($|\mathsf{supp}(i)|$ is the number of subscribers who have rated movie i), ρ_i and d_i are the rating and date, respectively, of movie i in the auxiliary information, and ρ_i' and d_i' are the rating and date in the candidate record r'.[5] As explained in section 6.4, this scoring function was chosen to favor statistically unlikely matches and thus minimize accidental false positives. The parameters ρ_0 and d_0 are 1.5 and 30 days, respectively. These were chosen heuristically, as they gave the best results in our experiments,[6] and used throughout, regardless of the amount of noise in Aux. The eccentricity parameter was set to $\phi = 1.5$, *i.e.*, the algorithm declares there is no match if and only if the difference between the highest and the second highest scores is no more than 1.5 times the standard deviation. (A constant value of ϕ does not always give the equal error rate, but it is a close enough approximation.)

Didn't Netflix publish only a sample of the data? Because Netflix published less than $\frac{1}{10}$ of its 2005 database, we need to be concerned about false positives. What if the adversary finds a record matching his aux in the published sample, but this is a false match and the real record has not been released at all?

Algorithm Scoreboard-RH is specifically designed to detect when the

[5] $\mathsf{wt}(i)$ is undefined when $|\mathsf{supp}(i)| = 0$, but this is not a concern since every movie is rated by at least 4 subscribers.

[6] It may seem that tuning the parameters to the specific dataset may have unfairly improved our results, but an actual adversary would have performed the same tuning. We do not claim that these numerical parameters should be used for other instances of our algorithm; they must be derived by trial and error for each target dataset.

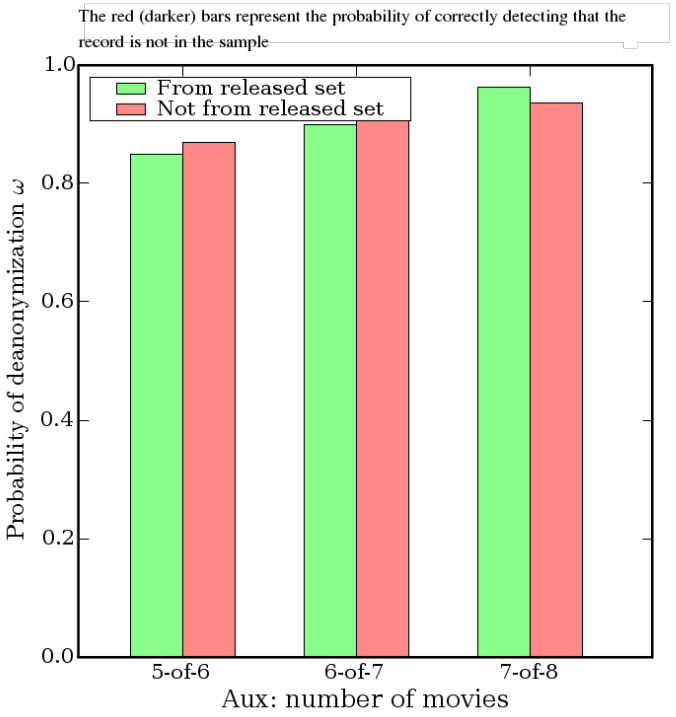

Figure 6.5: Same parameters as Fig. 6.4, but the adversary must also detect when the target record is not in the sample.

record corresponding to aux is *not* in the sample. We ran the following experiment. First, we gave aux from a random record to the algorithm and ran it on the dataset. Then we *removed* this record from the dataset and re-ran the algorithm. In the former case, the algorithm should find the record; in the latter, declare that it is not in the dataset. As shown in Fig. 6.5, the algorithm succeeds with high probability in both cases.

It is possible, although *extremely* unlikely, that the original Netflix dataset is not as sparse as the published sample, *i.e.*, it contains clusters of records which are close to each other, but only one representative of each cluster has been released in the Prize dataset. A dataset with such a structure would be exceptionally unusual and theoretically problematic (see Theorem 8).

If the amount of auxiliary information available to the adversary is less than shown in Fig. 6.5, the absence of false positives cannot be guaranteed *a priori*, but there is a lot of additional information in the dataset that can be used to eliminate them. For example, if the start date and the total number of movies in a record are part of the auxiliary information (*e.g.*, the adversary knows approximately when his target first joined Netflix), they can be used to eliminate candidate records.

Results of de-anonymization. We carried out the experiments summarized in Table 6.1.

Our conclusion is that very little auxiliary information is needed for de-anonymize an average subscriber record from the Netflix Prize dataset. With 8 movie ratings (of which 2 may be completely wrong) and dates that may have a 14-day error, 99% of records can be uniquely identified in the dataset. For 68%, *two* ratings and dates (with a 3-day error) are sufficient

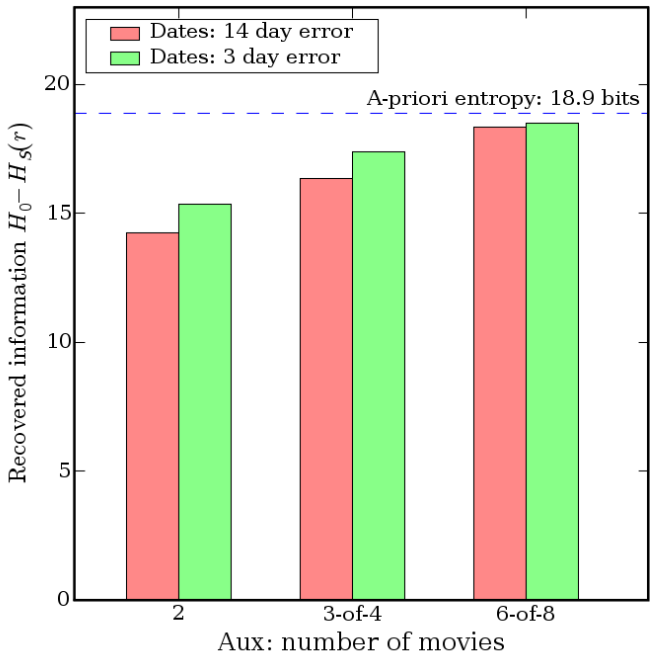

Figure 6.6: Entropic de-anonymization: same parameters as in Fig. 6.4.

Table 6.1: Summary of results

Fig	Ratings	Dates	Type	Aux selection
6.4	Exact	$\pm 3/\pm 14$	Best-guess	Uniform
6.5	Exact	$\pm 3/\pm 14$	Best-guess	Uniform
6.6	Exact	$\pm 3/\pm 14$	Entropic	Uniform
6.8	Exact	No info.	Best-guess	Not 100/500
6.9	± 1	± 14	Best-guess	Uniform
6.10	± 1	± 14	Best-guess	Uniform
6.11	Exact	No info.	Entropic	Not 100/500

(Fig. 6.4). Even for the other 32%, the number of possible candidates is brought down dramatically. In terms of entropy, the additional information required for complete de-anonymization is around 3 bits in the latter case (with no auxiliary information, this number is 19 bits). When the adversary knows 6 movies correctly and 2 incorrectly, the extra information he needs for complete de-anonymization is a fraction of a bit (Fig. 6.6).

Even without any dates, a substantial privacy breach occurs, especially when the auxiliary information consists of movies that are not blockbusters. In Fig. 6.7, we demonstrate how much information the adversary gains about his target just from the knowledge that the target watched a particular movie as a function of the rank of the movie.[7] Because there are correlations between the lists of subscribers who watched various movies, we cannot simply multiply the information gain per movie by the number of movies. Therefore, Fig. 6.7 cannot be used to infer how many movies the adversary needs to know for successful de-anonymization.

As shown in Fig. 6.8, two movies are no longer sufficient for de-anonymization, but 84% of subscribers present in the dataset can be uniquely identified if the adversary knows 6 out of 8 moves outside the top 500.

To show that this is not a significant limitation, consider that most subscribers rate fairly rare movies (Table 6.2).

Fig. 6.9 shows that the effect of relative popularity of movies known to the adversary is not dramatic.

In Fig. 6.10, we isolate the effect of adding more noise to the auxiliary information.

[7]We measure the rank of a movie by the number of subscribers who have rated it.

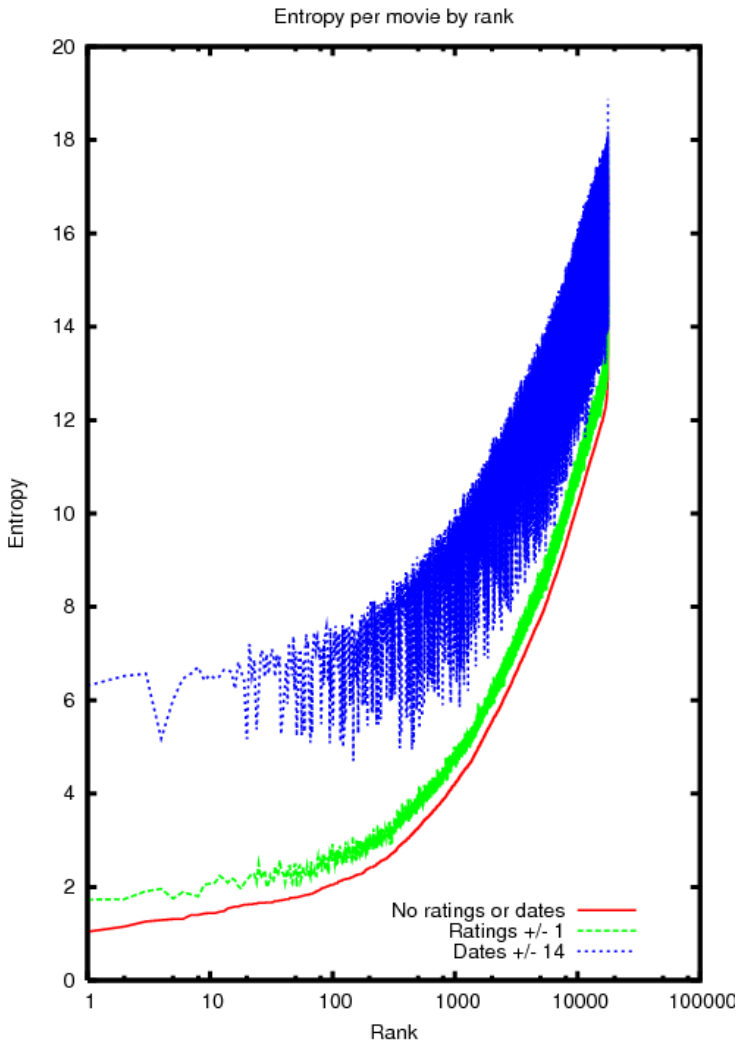

Figure 6.7: Entropy of movie by rank

Table 6.2: Subscribers rate rare movies

Not in X most rated	% of subscribers who rated ...		
	≥ 1 movie	≥ 5	≥ 10
$X = 100$	100%	97%	93%
$X = 500$	99%	90%	80%
$X = 1000$	97%	83%	70%

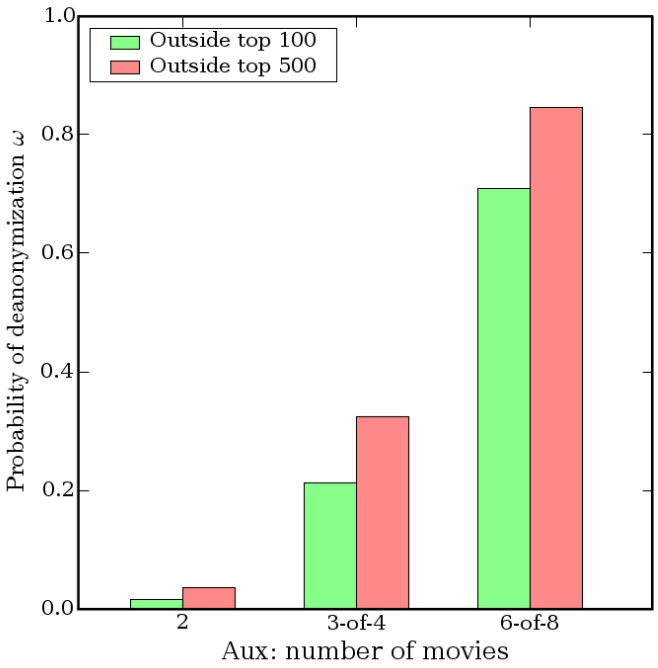

Figure 6.8: Adversary knows exact ratings but does not know dates at all.

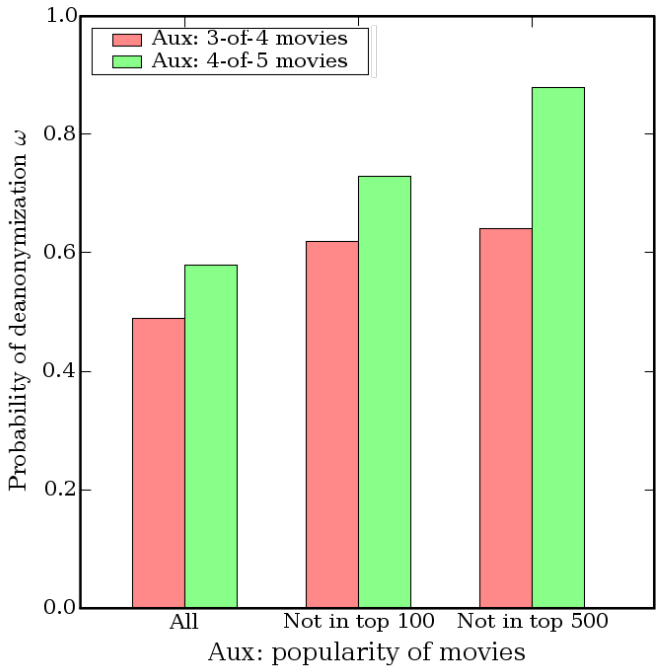

Figure 6.9: Effect of knowing less popular movies rated by victim. Adversary knows approximate ratings (± 1) and dates (14-day error).

Fig. 6.11 shows that even when the adversary's probability to correctly learn the attributes of the target record is low, he gains a lot of information about the target record. Even in the worst scenario, the additional information needed to to complete the de-anonymization has been reduced to less than half of its original value.

Fig. 6.12 shows why even partial de-anonymization can be very dangerous. There are many things the adversary might know about his target that are not captured by our formal model, such as the approximate number of movies rated, the date when they joined Netflix and so on. Once a candidate set of records is available, further automated analysis or human inspection might

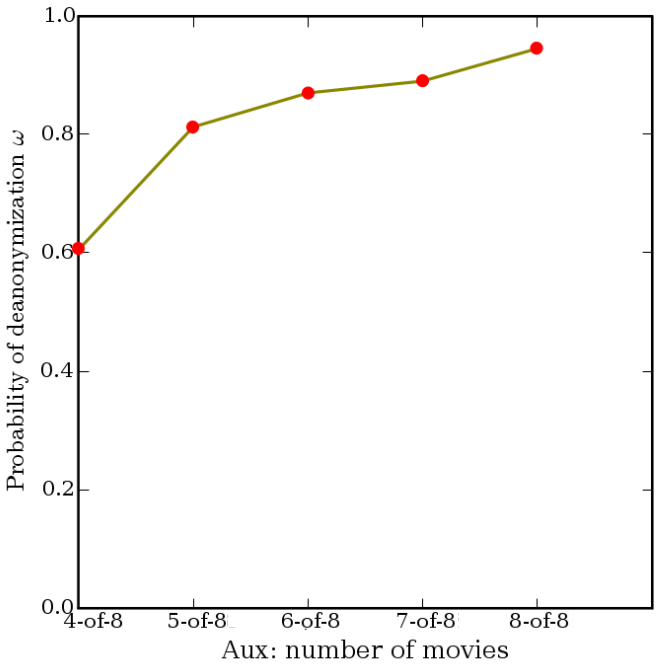

Figure 6.10: Effect of increasing error in Aux.

be sufficient to complete the de-anonymization. Fig. 6.12 shows that in some cases, knowing the number of movies the target has rated (even with a 50% error!) can more than double the probability of complete de-anonymization.

Obtaining the auxiliary information and IMDb cross-correlation. Given how little auxiliary information is needed to de-anonymize the average subscriber record from the Netflix Prize dataset, a determined adversary who targets a specific individual may not find it difficult to obtain such information, especially since it need not be precise. We emphasize that massive collection of data on thousands of subscribers is not the only or even the most important threat. A water-cooler conversation with an office colleague about her cinematographic likes and dislikes may yield enough information,

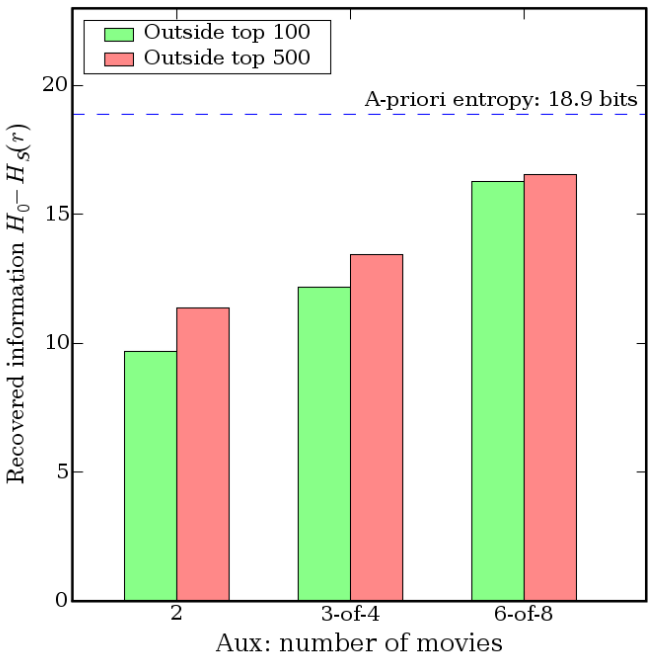

Figure 6.11: Entropic de-anonymization: same parameters as in Fig. 6.6.

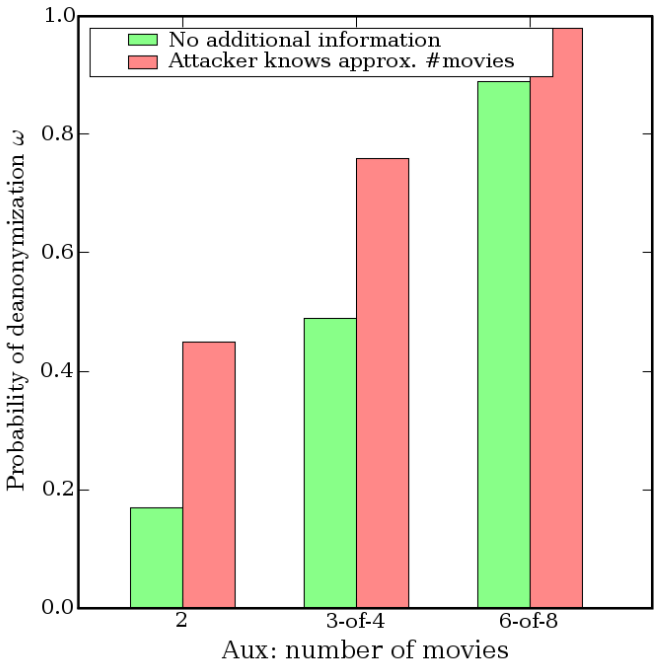

Figure 6.12: Effect of knowing approximate number of movies rated by victim (±50%). Adversary knows approximate ratings (±1) and dates (14-day error).

especially if at least a few of the movies mentioned are outside the top 100 most rated Netflix movies. This information can also be gleaned from personal blogs, Google searches, and so on.

One possible source of a large number of personal movie ratings is the Internet Movie Database (IMDb) [137]. We expect that for Netflix subscribers who use IMDb, there is a strong correlation between their private Netflix ratings and their public IMDb ratings.[8] Our attack does not require that all movies rated by the subscriber in the Netflix system be also rated in IMDb, or vice versa. In many cases, even a handful of movies that are rated by a

[8]We are *not* claiming that a large fraction of Netflix subscribers use IMDb, or that many IMDb users use Netflix.

subscriber in both services would be sufficient to identify his or her record in the Netflix Prize dataset (if present among the released records) with enough statistical confidence to rule out the possibility of a false match except for a negligible probability.

Due to the restrictions on crawling IMDb imposed by IMDb's terms of service (of course, a real adversary may not comply with these restrictions), we worked with a very small sample of around 50 IMDb users. Our results should thus be viewed as a proof of concept. They do not imply anything about the percentage of IMDb users who can be identified in the Netflix Prize dataset.

The auxiliary information obtained from IMDb is quite noisy. First, a significant fraction of the movies rated on IMDb are not in Netflix, and vice versa, *e.g.*, movies that have not been released in the US. Second, some of the ratings on IMDb are missing (*i.e.*, the user entered only a comment, not a numerical rating). Such data is still useful for de-anonymization because an average user has rated only a tiny fraction of all movies, so the mere fact that a person has watched a given movie tremendously reduces the number of anonymous Netflix records that could possibly belong to that user. Finally, IMDb users among Netflix subscribers fall into a continuum of categories with respect to rating dates, separated by two extremes: some meticulously rate movies on both IMDb and Netflix at the same time, and others rate them whenever they have free time (which means the dates may not be correlated at all). Somewhat offsetting these disadvantages is the fact that we can use all of the user's ratings publicly available on IMDb.

Because we have no "oracle" to tell us whether the record our algorithm has found in the Netflix Prize dataset based on the ratings of some IMDb user

indeed belongs to that user, we need to guarantee a very low false positive rate. Given our small sample of IMDb users, our algorithm identified the records of two users in the Netflix Prize dataset with eccentricities of around 28 and 15, respectively. These are exceptionally strong matches, which are highly unlikely to be false positives: the records in questions are **28 standard deviations** (respectively, 15 standard deviations) away from the second-best candidate. Interestingly, the first user was de-anonymized mainly from the ratings and the second mainly from the dates. For nearly all the other IMDb users we tested, the eccentricity was no more than 2.

Let us summarize what our algorithm achieves. Given a user's *public* IMDb ratings, which the user posted voluntarily to reveal *some* of his (or her; but we'll use the male pronoun without loss of generality) movie likes and dislikes, we discover *all* ratings that he entered *privately* into the Netflix system. Why would someone who rates movies on IMDb—often under his or her real name—care about privacy of his Netflix ratings? Consider the information that we have been able to deduce by locating one of these users' entire movie viewing history in the Netflix Prize dataset and that *cannot* be deduced from his public IMDb ratings.

First, his political orientation may be revealed by his strong opinions about "Power and Terror: Noam Chomsky in Our Times" and "Fahrenheit 9/11," and his religious views by his ratings on "Jesus of Nazareth" and "The Gospel of John." Even though one should not make inferences solely from someone's movie preferences, in many workplaces and social settings opinions about movies with predominantly gay themes such as "Bent" and "Queer as folk" (both present and rated in this person's Netflix record) would be considered sensitive. In any case, it should be for the individual and not for

Netflix to decide whether to reveal them publicly.

6.6 Summary

We have presented a de-anonymization methodology for sparse micro-data, and demonstrated its practical applicability by showing how to de-anonymize movie viewing records released in the Netflix Prize dataset. Our de-anonymization algorithm Scoreboard-RH works under very general assumptions about the distribution from which the data is drawn, and is robust to data perturbation and mistakes in the adversary's knowledge. Therefore, we expect that it can be successfully used against any dataset containing anonymous multi-dimensional records such as individual transactions, preferences, and so on.

We conjecture that the amount of perturbation that must be applied to the data to defeat our algorithm will completely destroy their utility for collaborative filtering. Sanitization techniques from the k-anonymity literature such as generalization and suppression do not provide meaningful privacy guarantees, and in any case fail on high-dimensional data. Furthermore, for most records simply knowing *which* columns are non-null reveals as much information as knowing the specific values of these columns. Therefore, any sanitization which leaves sensitive attributes untouched, as is the case for suppression and generalization [241, 62, 172], does not help.

Other possible countermeasures include interactive mechanisms for privacy-protecting data mining such as [35, 83], as well as more recent non-interactive techniques [36]. Both support only limited classes of computations such as statistical queries and learning halfspaces. By contrast, in scenarios such as the Netflix Prize, the purpose of the data release is precisely to foster compu-

tations on the data that have not even been foreseen at the time of release [9], and are vastly more sophisticated than the computations that we know how to perform in a privacy-preserving manner.

An intriguing possibility was suggested by Matthew Wright via personal communication: to release the records without the column identifiers (*i.e.*, movie names in the case of the Netflix Prize dataset). It is not clear how much worse the current data mining algorithms would perform under this restriction. Furthermore, this does not appear to make de-anonymization impossible, but merely harder. Nevertheless, it is an interesting countermeasure to investigate.

McSherry and Mironov have recently developed a technique for providing provable privacy guarantees in collaborative filtering based on a relaxed definition of differential privacy [178]. While not perfect, it is a way to protect privacy without relying on anonymity. Depending on the context, the best defense against de-anonymization algorithms might be to sidestep the anonymity question altogether.

[9] As of the current writing, the best algorithm in the Netflix Prize competition is a combination of 107 different techniques.

Chapter 7

De-anonymizing Social Networks

7.1 Introduction

Social networks have been studied for a century [229] and are a staple of research in disciplines such as epidemiology [24], sociology [248, 119, 37], economics [120], and many others [80, 26, 127]. The recent proliferation of online social networks such as MySpace, Facebook, Twitter, and so on has attracted attention of computer scientists, as well [152].

Even in the few online networks that are completely open, there is a disconnect between users' willingness to share information and their reaction to unintended parties viewing or using this information [55]. Most operators thus provide at least some privacy controls. Many online and virtually all offline networks (*e.g.*, telephone calls, email and instant messages, *etc.*) restrict access to the information about individual members and their relationships.

Network owners often share this information with advertising partners and other third parties. Such sharing is the foundation of the business case for many online social-network operators. Some networks are even published for research purposes. To alleviate privacy concerns, the networks are *anonymized*, *i.e.*, names and demographic information associated with individual nodes are suppressed. Such suppression is often misinterpreted as removal of "personally identifiable information" (PII), even though PII may include much more than names and identifiers (see the discussion in Section 7.3.4). For example, the EU privacy directive defines "personal data" as "any information relating to an identified or identifiable natural person [...]; an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity" [88].

Anonymity has been unquestioningly interpreted as equivalent to privacy in several high-profile cases of data sharing. After a New York court ruling ordering Google to hand over viewing data of over 100 million YouTube users to Viacom and the subsequent protests from privacy advocates, a revised agreement was struck under which Google would anonymize the data before handing it over [242]. The CEO of NebuAd, a U.S. company that offers targeted advertising based on browsing histories gathered from ISPs, dismissed privacy concerns by saying that "We don't have any raw data on the identifiable individual. Everything is anonymous" [63]. Phorm, a U.K. company with a similar business model, aims to collect the data on Web-surfing habits of 70% of British broadband users; the only privacy protection is that user identities are mapped to random identifiers [235]. In social networks, too, user anonymity has been used as the answer to all privacy concerns (see Sec-

tion 7.2).

Our contributions. This is the first work to demonstrate feasibility of large-scale, passive de-anonymization of real-world social networks.

First, we survey the current state of data sharing in social networks, the intended purpose of each type of sharing, the resulting privacy risks, and the wide availability of auxiliary information which can aid the attacker in de-anonymization.

Second, we formally define privacy in social networks and relate it to node anonymity. We identify several categories of attacks, differentiated by attackers' resources and auxiliary information. We also give a methodology for measuring the extent of privacy breaches in social networks, which is an interesting problem in its own right.

Third, we develop a generic re-identification algorithm for anonymized social networks. The algorithm uses only the network structure, does not make any *a priori* assumptions about membership overlap between multiple networks, and defeats all known defenses.

Fourth, we give a concrete demonstration of how our de-anonymization algorithm works by applying it to Flickr and Twitter, two large, real-world online social networks. We show that a third of the users who are verifiable members of both Flickr and Twitter[1] can be recognized in the completely anonymous Twitter graph with only 12% error rate, even though the overlap in the relationships for these members is less than 15%!

Sharing of anonymized social-network data is widespread and the auxiliary information needed for our attack is commonly available. We argue that our work calls for a substantial re-evaluation of business practices surrounding

[1] At the time of our crawl; details are in Section 7.6.

the sharing of social-network data.

7.2 State of the Union

The attacks described in this work target anonymized, sanitized versions of social networks, using partial auxiliary information about a subset of their members. To show that both anonymized networks and auxiliary information are widely available, we survey real-world examples of social-network data sharing, most of which involve releasing *more* information than needed for our attack.

Academic and government data-mining. Social networks used for published data-mining research include the mobile-phone call graphs of, respectively, 7 million [202], 3 million [190], and 2.5 million [157] customers, as well as the land-line phone graph of 2.1 million Hungarian users [154]. Corporations like AT&T, whose own database of 1.9 trillion phone calls goes back decades [133], have in-house research facilities, but smaller operators must share their graphs with external researchers. Phone-call networks are also commonly used to detect illicit activity such as calling fraud [260] and for national security purposes, such as identifying the command-and-control structures of terrorist cells by their idiosyncratic sub-network topologies [133]. A number of companies sell data-mining solutions to governments for this purpose [231].

Sociologists, epidemiologists, and health-care professionals collect data about geographic, friendship, family, and sexual networks to study disease propagation and risk. For example, the Add Health dataset includes the sexual-relationship network of almost 1,000 students of an anonymous Midwestern high school as part of a detailed survey on adolescent health [8]. While

the Add Health project takes a relatively enlightened stance on privacy [7], this graph has been published in an anonymized form [28].

For online social networks, the data can be collected by crawling either via an API, or "screen-scraping" (*e.g.*, Mislove *et al.* crawled Flickr, YouTube, LiveJournal, and Orkut [184]; anonymized graphs are available by request only). We stress that even when obtained from public websites, this kind of information—if publicly released—still presents privacy risks because it helps attackers who lack resources for massive crawls. In some online networks, such as LiveJournal and the Experience Project, user profiles and relationship data are public, but many users maintain pseudonymous profiles. From the attacker's perspective, this is the same as publishing the anonymized network.

Advertising. With the emergence of concrete evidence that social-network data makes commerce much more profitable [218, 238], network operators are increasingly sharing their graphs with advertising partners to enable better social targeting of advertisements. For example, Facebook explicitly says that users' profiles may be shared for the purpose of personalizing advertisements and promotions, as long as the individual is not explicitly identified [90]. Both Facebook and MySpace allow advertisers to use friends' profile data for ad targeting [70]. Social-network-driven advertising has been pursued by many startups [85, 189] and even Google [221], typically relying on anonymity to prevent privacy breaches [18, 86, 201].

Third-party applications. The number of third-party applications on Facebook alone is in the tens of thousands and rapidly growing [222]. The data from multiple applications can be aggregated and used for targeted advertising (*e.g.*, as done by SocialMedia [217]). As the notion of social networking

as a feature rather than destination takes hold [17], many other networks are trying to attract application developers; on the Ning platform, which claims over 275,000 networks, each network can be considered a third-party application. The data given to third-party applications is usually not anonymized, even though most applications would be able to function on anonymized profiles [94].

Third-party applications have a poor track record of respecting privacy policies. For example, a security hole in a Facebook application developed by Slide, Inc. "exposed the birthdays, gender, and relationship status of strangers, including Facebook executives, [and] the wife of Google co-founder Larry Page" [183]. WidgetLaboratory, one of the most popular developers for the Ning platform, was banned permanently after "gathering credentials from users and otherwise creating havoc on Ning networks" [20]. Therefore, it is important to understand what a malicious third-party application can learn about members of a social network, even if it obtains the data in an anonymized form.

Aggregation. Aggregation of information from multiple social networks, facilitated by projects such as OpenID [3], DataPortability [71], the "social graph" project [97], and various microformats [2], potentially presents a greater threat to individual privacy than one-time data releases. Existing aggregators include FriendFeed, MyBlogLog, Jaiku (recently acquired by Google), and Plaxo; the latter even provides an open-source "social graph crawler" [211]. Aggregated networks are an excellent source of auxiliary information for our attacks.

Other data-release scenarios. WellNet is a health-care co-ordination ser-

vice which enables employers to monitor the social network in real time in order to track employees' medical and pharmacy activity [180]. The data is anonymized.

In "friend-to-friend networking," a peer-to-peer file-sharing network is overlaid on social links [213] in order to defeat censor nodes such as the RIAA. Nodes are pseudonymous and communication is encrypted. Since traffic is typically not anonymized at the network level, the logs that can be obtained, for example, by subpoenaing the ISP are essentially anonymized social-network graphs.

Finally, consider photographs published online without identifying information. The accuracy of face recognition can be improved substantially by exploiting the fact that users who appear together in photographs are likely to be neighbors in the social network [234]. Since most online photographs appear in a social-network context, they effectively represent an anonymized graph, and techniques developed in this work can help in large-scale facial re-identification.

7.3 Background and related Work

7.3.1 Privacy properties.

A social network consists of nodes, edges, and information associated with each node and edge. The existence of an edge between two nodes can be sensitive: for instance, in a sexual-relationship network with gender information attached to nodes [28] it can reveal sexual orientation. *Edge privacy* was considered in [151, 23]. In most online social networks, however, edges are public by

default, and few users change the default settings [123].

While the mere presence of an edge may not be sensitive, edge attributes may reveal more information (*e.g.*, a single phone call vs. a pattern of calls indicative of a business or romantic relationship). For example, phone-call patterns of the disgraced NBA referee Tom Donaghy have been used in the investigation [261]. In online networks such as LiveJournal, there is much variability in the semantics of edge relationships [98].

The attributes attached to nodes, such as the user's interests, are usually far more sensitive. Social Security numbers can be predicted from Facebook profiles with higher accuracy than random guessing [123]; see [57] for other privacy breaches based on profile data. Even implicit attributes such as node degree can be highly sensitive, *e.g.*, in a sexual network [28]. Existing defenses focus on names and other identifiers, but basic de-anonymization only reveals that someone belongs to the network, which is hardly sensitive. As we show in the rest of this chapter, however, it can be used as a vehicle for more serious attacks on privacy, including disclosure of sensitive attributes.

7.3.2 De-anonymization attacks.

Backstrom *et al.* present two *active* attacks on edge privacy in anonymized social networks [23]. These active attacks fundamentally assume that the adversary is able to modify the network prior to its release: "an adversary chooses an arbitrary set of users whose privacy it wishes to violate, creates a small number of new user accounts with edges to these targeted users, and creates a pattern of links among the new accounts with the goal of making it stand out in the anonymized graph structure." Both attacks involve creating

$O(\log N)$ new "sybil" nodes (N is the total number of nodes), whose outgoing edges help re-identify quadratically as many existing nodes.

Active attacks are difficult to stage on a large scale. First, they are restricted to online social networks (OSNs); creating thousands of fake nodes in a phone-call or real-life network is prohibitively expensive or impossible. Even in OSNs, many operators (*e.g.*, Facebook) check the uniqueness of email addresses and deploy other methods for verifying accuracy of supplied information, making creation of a large number of dummy nodes relatively difficult.

Second, the attacker has little control over the edges *incoming* to the nodes he creates. Because most legitimate users will have no reason to link back to the sybil nodes, a subgraph with no incoming edges but many outgoing edges will stand out. As we show below, this may enable the network operator to recognize that the network has been compromised by a sybil attack. There are also other techniques for identifying sybil attacks in social networks [267], including methods for spammer detection deployed by OSNs that allow unidirectional edges [225].

We carried out an experiment to verify the claim that identification of subgraphs consisting primarily of sybil nodes is difficult in real-world social networks. The data for this experiment was the graph of LiveJournal obtained from Mislove *et al.* [184], crawled in late 2006. It is a directed graph with 5.3 million nodes and 77 million edges. Except for the time of the crawl, this graph is similar to that used in [23].

The cut-based attack of [23] creates 7-node subgraphs containing a Hamiltonian path. In contrast to the observation in [23] that every possible 7-node subgraph containing a Hamiltonian path occurs in the LiveJournal graph, there are no subgraphs in the LiveJournal graph that have these two

properties and, furthermore, do not have any incoming edges. We conclude that active attacks are easy to detect if real users never link back to sybil nodes. More sophisticated sybil-detection techniques may work as long as only a small percentage of real users link back to sybil nodes.

The third limitation of active attacks is the fact that many OSNs require a link to be mutual before the information is made available in any form. Therefore, assuming that real users do not link back to dummy users, the links from fake nodes to real ones do not show up in the network.

We conclude that large-scale active attacks requiring creation of tens of thousands of sybil nodes are unlikely to be feasible. Active attacks can still be useful in identifying or creating a small set of "seeds" to serve as a starting point for large-scale, *passive* privacy breaches. We develop such an attack in Section 7.5.2.

Backstrom *et al.* also describe passive attacks, in which a small coalition of users discover their location in the anonymized graph by utilizing the knowledge of the network structure around them. This attack is realistic, but again, only works on a small scale: the colluding users can only compromise the privacy of some of the users who are already their friends.

By contrast, our attack does not require creation of a large number of sybil nodes, and—as shown by our experiments on real-world online social networks—can be successfully deployed on a very large scale.

7.3.3 Defenses.

Existing privacy protection mechanisms for social networks are only effective against very restricted adversaries and have been evaluated on small, simulated

networks whose characteristics are different from real social networks. For example, Zheleva and Getoor give several strategies for preventing link re-identification [268], but the model ignores auxiliary information that may be available to the attacker.

An unusual attempt to prevent network operators from capitalizing on user-provided data appears in [125]. It involves scrambling the profiles when they are sent to the server and client-side unscrambling when a friend's profile is viewed. Building and running such a system involves constant reverse-engineering of communication between the client and the server. Further, all of a user's friends need to use the system, flatly contradicting the claim of incremental deployability. A similar idea appears in [168], with a more sound architecture based on a server-side Facebook application. Both approaches severely cripple social-network functionality because almost any non-trivial action other than viewing another user's profile or messages requires the server to manipulate the data in a way which is not possible under encryption.

Anonymity is a popular approach to protecting privacy. Felt and Evans propose a system where applications see randomized tokens representing users instead of actual identifiers [94]. Frikken and Golle show how to compute an anonymous graph from pieces held by different participants in order to perform privacy-preserving social-network analysis [105]. Kerschbaum and Schaad additionally enable participants to track their position in the anonymous graph [148].

Several papers proposed variants of k-anonymity for social networks. For example, Hay *et al.*[132] analyze "k-candidate anonymity" under sanitization through random edge perturbation of the edges. Anonymity is achieved only against severely restricted adversaries: in one model, the attacker only has

information about degree sequences around his target node; in another, partial knowledge of the structure in the vicinity of the target. The technique appears to work only if the average degree is low, ruling out most online social networks. A follow-up paper [131] considers "graph generalization" which clusters nodes into "super-nodes." While this technique protects against re-identification, it appears to render the graph useless for the applications considered in the previous subsection.

Liu and Terzi consider node re-identification assuming that the adversary's auxiliary information consists only of node degrees [167]. There is no clear motivation for this restriction. Campan and Truta propose metrics for the information loss caused by edge addition and deletion and apply k-anonymity to node attributes as well as neighborhood structure [51]. Zhou and Pei assume that the adversary knows the exact 1-neighborhood of the target node [269]. The anonymization algorithm attempts to make this 1-neighborhood isomorphic to $k-1$ other 1-neighborhoods via edge addition. The experiments are performed on an undirected network with average degree 4 (an order of magnitude lower than that in real social networks) and already require increasing the number of edges by 6%. The number of edges to be added and the computational effort are likely to rise sharply with the average degree.

The fundamental problem with k-anonymity is that it is a syntactic property which may not provide any privacy even when satisfied (*e.g.*, if all k isomorphic neighborhoods have the same value of some sensitive attributes). Crucially, all of these defenses impose arbitrary restrictions on the information available to the adversary and make arbitrary assumptions about the properties of the social network.

We argue that the auxiliary information which is likely to be available

to the attacker is *global* in nature (*e.g.*, another social network with partially overlapping membership) and not restricted to the neighborhood of a single node. In the rest of this chapter, we show how this information, even if very noisy, can be used for large-scale re-identification. Existing models fail to capture self-reinforcing, feedback-based attacks, in which re-identification of some nodes provides the attacker with more auxiliary information, which is then used for further re-identification. Development of a model for such attacks is our primary contribution.

7.3.4 On "Personally Identifiable Information"

"Personally identifiable information" is a legal term used in two related but distinct contexts. The first context is a series of breach-disclosure laws enacted in recent years in response to security breaches involving customer data that could enable identity theft.

California Senate Bill 1386 [49] is a representative example. It defines "personal information" as follows:

> [An] individual's first name or first initial and last name in combination with any one or more of the following data elements, when either the name or the data elements are not encrypted:
>
> - Social security number.
> - Driver's license number or California Identification Card number.
> - Account number, credit or debit card number, in combination with any required security code, access code, or password that would permit access to an individual's financial account.

Two points are worthy of note. First, the spirit of the terminology is to capture the types of information that are commonly used for authenticating an individual. This reflects the bill's intent to deter identity theft. Consequently, data such as email addresses and telephone numbers do not fall under the scope of this law. Second, it is the personal information itself that is sensitive, rather than the fact that it is possible to associate sensitive information with an identity.

The second context in which the term "personally identifiable information" appears is the privacy law. In the United States, the Privacy Act of 1974 [249] regulates the collection of personal information by government agencies, but there is no overarching law regulating private entities. At least three such acts introduced in 2005 failed to pass: the Privacy Act of 2005 [254], the Consumer Privacy Protection Act of 2005 [252], and the Online Privacy Protection Act of 2005 [253]. However, there do exist laws for specific types of data such as the Video Privacy Protection Act (VPPA) [87] and the Health Insurance Privacy and Accountability Act (HIPAA).

The language from the HIPAA Privacy Rule [250] is representative:

> Individually identifiable health information is information
>
> [. . .]
>
> 1. That identifies the individual; or
> 2. With respect to which there is a reasonable basis to believe the information can be used to identify the individual.

The spirit of the law clearly encompasses deductive disclosure, and the term "reasonable basis" leaves the defining line open to interpretation by case

law. We are not aware of any court decisions that define identifiability.

Individual U.S. states do have privacy protection laws that apply to any operator, such as California's Online Privacy Protection Act of 2003 [50]. Some countries other than the United States have similar generic laws, such as Canada's Personal Information Protection and Electronic Documents Act (PIPEDA) [206]. The European Union is notorious for the broad scope and strict enforcement of its privacy laws—the EU privacy directive defines "personal data" as follows [88]:

> any information relating to an identified or identifiable natural person [...]; an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity.

It is clear from the above that privacy law, as opposed to breach-disclosure law, in general interprets personally identifiable information broadly, in a way that is not covered by syntactic anonymization. This distinction appears to be almost universally lost on companies that collect and share personal information, as illustrated by the following Senate Committee testimony by Chris Kelly, Chief Privacy Officer of Facebook [144]:

> The critical distinction that we embrace in our policies and practices, and that we want our users to understand, is between the use of personal information for advertisements in personally-identifiable form, and the use, dissemination, or sharing of information with advertisers in non-personally-identifiable form. Ad targeting that

> shares or sells personal information to advertisers (name, email, other contact oriented information) without user control is fundamentally different from targeting that only gives advertisers the ability to present their ads based on aggregate data.

Finally, it is important to understand that the term "personally identifiable information" has no particular technical meaning. Algorithms that can identify a user in an anonymized dataset are agnostic to the semantics of the data elements. While some data elements may be uniquely identifying on their own, *any* element can be identifying in combination with others. The feasibility of such re-identification has been amply demonstrated by the AOL privacy fiasco [129], de-anonymization of the Netflix Prize dataset [194], and the work presented in this paper. It is regrettable that the mistaken dichotomy between personally identifying and non-personally identifying attributes has crept into the technical literature in phrases such as "quasi-identifier."

7.4 Model and Definitions

7.4.1 Social network

A social network $\mathcal{S}$ consists of (1) a directed graph $G = (V, E)$, and (2) a set of attributes $\mathcal{X}$ for each node in V (for instance, name, telephone number, *etc.*) and a set of attributes $\mathcal{Y}$ for each edge in E (for instance, type of relationship). The model is agnostic as to whether attributes accurately reflect real-world identities or not (see below). We treat attributes as atomic values from a discrete domain; this is important for our formal definition of privacy breach (Definition 10 below). Real-valued attributes must be discretized. Where

specified, we will also represent edges as attributes in $\mathcal{Y}$ taking values in $\{0, 1\}$.

In addition to the explicit attributes, some privacy policies may be concerned with implicit attributes, *i.e.*, properties of a node or an edge that are based purely on the graph structure. For example, node degree can be a sensitive implicit attribute. Implicit attributes may be leaked without disclosing any explicit attributes. For example, if the adversary re-identifies a subset of nodes in an anonymized graph, none of which are adjacent, he learns the degrees of these nodes without breaking edge privacy. Which implicit attributes should be protected depends on the specific network.

7.4.2 "Identity" in social networks

The correspondence between accounts or profiles (*i.e.*, network nodes) and real-world identities varies greatly from social network to social network. A wired telephone may be shared by a family or an office, while mobile phones are much more likely to belong to a single person. Some online social networks such as Facebook attempt to ensure that accounts accurately reflect real-world information [244], while others such as MySpace are notoriously lax [171]. Fake MySpace profiles have been created for pets and celebrities, and a user may create multiple profiles with contradictory or fake information.

In this paper, we eschew an explicit notion of identity and focus instead on *entities*, which are simply sources of social-network profile information that are consistent across different networks and service providers. In most cases, an entity is associated with a real-world person, but does not have to be (*e.g.*, consider a political campaign which has a YouTube account and a Twitter account). The concept of entities also allows us to capture information which

is characteristic of a user across multiple networks—for example, an unusual username—but is not related to anything in the real world.

In our model, nodes are purely collections of their attributes, and to *identify* a node simply means to learn the entity to which the node belongs, whether this entity is a single person, a group, or an organization. We assume that correctly associating a node with the corresponding entity constitutes a breach of anonymity. The question of whether the entity is a single individual or not is extraneous to our model.

7.4.3 Data release

Our model of the data release process focuses on what types of data are released and how the data is sanitized (if at all), and abstracts away from the procedural distinctions such as whether the data is available in bulk or obtained by crawling the network. As discussed in Section 7.2, social-network data are routinely released to advertisers, application developers, and researchers. Advertisers are often given access to the entire graph in a (presumably) anonymized form and a limited number of relevant attributes for each node. Application developers, in current practice, get access to a subgraph via user opt-in and most or all of the attributes within this subgraph. This typically includes the identifying attributes, even if they are not essential for the application's functionality [94]. Researchers may receive the entire graph or a subgraph (up to the discretion of the network owner) and a limited set of non-identifying attributes.

"Anonymization" is modeled by publishing only a subset of attributes. Unlike naïve approaches such as k-anonymity, we do not distinguish identifying

and non-identifying attributes (any attribute can be identifying if it happens to be known to the adversary as part of his auxiliary information). Suppressed attributes are not limited to the demographic quasi-identifiers *a priori*; we simply assume that the published attributes by themselves are insufficient for re-identification. In Section 7.4.5, we explain the (indirect) connection between preventing node re-identification and intuitive "privacy." In terms of entropy, most of the information in the released graph resides in the edges, and this is what our de-anonymization algorithm will exploit.

The data release process may involve perturbation or sanitization that changes the graph structure in some way to make re-identification attacks harder. As we argued in Section 7.3, deterministic methods that attempt to make different nodes look identical do not work on realistic networks. Other defenses are based on injecting random noise into the graph structure. The most promising one is *link prediction* [162], which produces plausible fake edges by exploiting the fact that edges in social-network graphs have a high clustering coefficient. (We stress that link prediction is far beyond the existing sanitization techniques, which mostly rely on simple removal of identifiers.) The experiments in Section 7.6.2 show that our algorithm is robust to injected noise, whether resulting from link prediction or not. In Appendix D, we discuss how to measure the amount of noise introduced by perturbation.

We model the data sanitization and release process as follows. First, select a subset of nodes, $V_{\mathsf{san}} \subset V$, and subsets $\mathcal{X}_{\mathsf{san}} \subseteq \mathcal{X}, \mathcal{Y}_{\mathsf{san}} \subseteq \mathcal{Y}$ of node and edge attributes to be released. Second, compute the induced subgraph on V_{san}. For simplicity, we do not model more complex criteria for releasing edge, *e.g.*, based on edge attributes. Third, remove some edges and add fake edges. Release $S_{\mathsf{san}} = (V_{\mathsf{san}}, E_{\mathsf{san}}, \{X(v) \forall v \in V_{\mathsf{san}}, X \in \mathcal{X}_{\mathsf{san}}\}, \{Y(e) \forall e \in E_{\mathsf{san}}, Y \in$

$\mathcal{Y}_{\mathsf{san}}\}$), *i.e.*, a sanitized subset of nodes and edges with the corresponding attributes.

7.4.4 Threat model

As described in Section 7.2, network owners release anonymized and possibly sanitized network graphs to commercial partners and academic researchers. Therefore, we take it for granted that the attacker will have access to such data. The main question we answer in the rest of this chapter is: **can sensitive information about specific individuals be extracted from anonymized social-network graphs?**

Attack scenarios. Attackers fall into different categories depending on their capabilities and goals. The strongest adversary is a government-level agency interested in *global surveillance*. Such an adversary can be assumed to already have access to a large auxiliary network S_{aux} (see below). His objective is large-scale collection of detailed information about as many individuals as possible. This involves aggregating the anonymous network S_{san} with S_{aux} by recognizing nodes that correspond to the same individuals.

Another attack scenario involves *abusive marketing*. A commercial enterprise, especially one specializing in behavioral ad targeting [247, 266], can easily obtain an anonymized social-network graph from the network operator for advertising purposes. As described in Sections 7.1 and 7.2, anonymity is often misinterpreted as privacy. If an unethical company were able to de-anonymize the graph using publicly available data, it could engage in abusive marketing aimed at specific individuals. *Phishing and spamming* also gain from social-network de-anonymization. Using detailed information about the

victim gleaned from his or her de-anonymized social-network profile, a phisher or a spammer will be able to craft a highly individualized, believable message (cf. [139]).

Yet another category of attacks involves *targeted de-anonymization* of specific individuals by stalkers, investigators, nosy colleagues, employers, or neighbors. In this scenario, the attacker has detailed contextual information about a single individual, which may include some of her attributes, a few of her social relationships, membership in other networks, and so on. The objective is to use this information to recognize the victim's node in the anonymized network and to learn sensitive information about her, including all of her social relationships in that network.

Modeling the attacker. We assume that in addition to the anonymized, sanitized target network S_{san}, the attacker also has access to a *different* network S_{aux} whose membership partially overlaps with S. The assumption that the attacker possesses such an auxiliary network is very realistic. First, it may be possible to extract S_{aux} directly from S: for example, parts of some online networks can be automatically crawled, or a malicious third-party application can provide information about the subgraph of users who installed it. Second, the attacker may collude with an operator of a different network whose membership overlaps with S. Third, the attacker may take advantage of several ongoing aggregation projects (see Section 7.2). The intent of these projects is benign, but they facilitate the creation of a global auxiliary network combining bits and pieces of public information about individuals and their relationships from multiple sources. Fourth, government-level aggregators, such as intelligence and law enforcement agencies, can collect data via surveillance and

court-authorized searches. Depending on the type of the attacker, the nodes of his auxiliary network may be a subset, a superset, or overlap with those of the target network.

We emphasize that even with access to a substantial auxiliary network S_{aux}, de-anonymizing the target network S_{san} is a highly non-trivial task. First, the overlap between the two networks may not be large. For the entities who are members of both S_{aux} and S, some social relationships may be preserved, *i.e.*, if two nodes are connected in S_{aux}, the corresponding nodes in S are also connected with a non-negligible probability, but many of the relationships in each network are unique to that network. Even if the same entity belongs to both networks, it is not immediately clear how to *recognize* that a certain anonymous node from S_{san} corresponds to the same entity as a given node from S_{aux}. Therefore, easy availability of auxiliary information does not directly imply that anonymized social networks are vulnerable to privacy breaches.

Our formal model of the attacker includes both aggregate auxiliary information (large-scale information from other data sources and social networks whose membership overlaps with the target network) and individual auxiliary information (identifiable details about a small number of individuals from the target network and possibly relationships between them). In the model, we consider edge relationship to be a binary attribute in $\mathcal{Y}$ and all edge attributes $Y \in \mathcal{Y}$ to be defined over V^2 instead of E. If $(u, v) \notin E$, then $Y[u, v] = \perp \ \forall Y \in \mathcal{Y}$.

Aggregate auxiliary information. It is essential that the attacker's auxiliary information may include relationships between entities. Therefore, we model S_{aux} as a graph $G_{\mathsf{aux}} = \{V_{\mathsf{aux}}, E_{\mathsf{aux}}\}$ and a set of probability distribu-

tions Aux_X and Aux_Y, one for each attribute of every node in V_{aux} and each attribute of every edge in E_{aux}. These distributions represent the adversary's (imperfect) knowledge of the corresponding attribute value. For example, the adversary may be 80% certain that an edge between two nodes is a "friendship" and 20% that it is a mere "contact." Since we treat edges themselves as attributes, this also captures the attacker's uncertain knowledge about the existence of individual edges. This model works well in practice, although it does not capture some types of auxiliary information, such as "node v_1 is connected to either node v_2, or node v_3."

For an attribute X of a node v (respectively, attribute Y of an edge e), we represent by $\mathsf{Aux}[X, v]$ (resp., $\mathsf{Aux}[Y, e]$) the attacker's prior probability distribution (*i.e.*, distribution given by his auxiliary information) of the attribute's value. The set Aux_X (resp., Aux_Y) can be thought of as a union of $\mathsf{Aux}[X, v]$ (resp., $\mathsf{Aux}[Y, e]$) over all attributes and nodes (resp., edges).

Aggregate auxiliary information is used in the the "propagation" stage of our de-anonymization algorithm (Section 7.5).

Individual auxiliary information (information about seeds). We also assume that the attacker possesses detailed information about a very small[2] number of members of the target network S. We assume that the attacker can determine if these members are also present in his auxiliary network S_{aux} (*e.g.*, by matching usernames and other contextual information). The privacy question is whether this information about a handful of members of S can be used, in combination with S_{aux}, to learn sensitive information about *other* members of S.

[2]Negligible relative to the size of S. For example, in our experiments, we find that between 30 and 150 seeds are sufficient for networks with 10^5 to 10^6 members.

It is not difficult to collect such data about a small number of nodes. If the attacker is already a user of S, he knows all details about his own node and its neighbors [151, 234]. Some networks permit manual access to profiles even if large-scale crawling is restricted (*e.g.*, Facebook allows viewing of information about "friends" of any member by default.) Some users may make their details public even in networks that keep them private by default. The attacker may even pay a handful of users for information about themselves and their friends [160], or learn it from compromised computers or stolen mobile phones. For example, the stored log of phone calls provides auxiliary information for de-anonymizing the phone-call graph. With an active attack (*e.g.*, [23]), the attacker may create fake nodes and edges in S with features that will be easy to recognize in the anonymized version of S, such as a clique or an almost-clique. Since large-scale active attacks are unlikely to be feasible (see Section 7.3.2), we restrict their role to collecting individual auxiliary information as a precursor to the main, passive attack.

Individual auxiliary information is used in the the "seed identification" stage of our de-anonymization algorithm (Section 7.5).

7.4.5 Breaching privacy

The fact that we are dealing with non-relational data makes it difficult to come up with a comprehensive definition of privacy in social networks. In general, one would like to say that properties of individual nodes should be privacy-sensitive and thus difficult to learn from the sanitized network, while aggregate properties should be learnable. But what counts as a "property of an individual node?" A natural candidate is any property about a k-neighborhood for some

small k (for instance, a property that a user has 3 different paths of length 2 to a known Al-Qaeda operative). Unfortunately, there does not seem to be an elegant way of choosing k because social-network graphs have a very small diameter due to the "six degrees of separation" phenomenon [248].

A related approach is differential privacy [81], which in the social-network context would require that the graph look roughly the same if any single node is removed. It is not obvious how to define node removal, and far from clear how to achieve differential privacy on graph-structured data, because aggregate properties of a graph can change substantially with the removal of a single node.

Even when the privacy policy is defined as a simple labeling of attributes (as we do below), the policy can be *global* or *granular*. With a global policy, the same privacy label applies to a given attribute in every node (*e.g.*, email addresses are either public for all members, or private for all members). Similarly, the edges in the network are either all public, or all private. With granular policies, the privacy setting can be different for each edge and each attribute of each node.

A global policy is sufficient most of the time. In most contexts, the network operator promises users that none of their data will be released in a personally identifiable way, implying a privacy policy where all edges and all attributes are private. In other contexts, some attributes might be intuitively understood to be public (*e.g.*, node degree) and others private.

Many online social-network services such as Facebook allow users to configure their individual privacy policy with a high level of granularity. This might become a common practice in the future, but so far it appears that the vast majority of users do not change their default settings [123, 153]. There is

also some ambiguity in modeling user preferences as formal privacy policies: for instance, an edge may be considered public by one endpoint and private by the other.

To keep the model simple and tractable, we do not use richer formalisms which may be suitable for some situations. For example, a multi-graph is a better model for social networks representing phone calls between individuals. We ignore the complex structure of node and edge attributes that may be relevant to privacy, such as "X knows Y through Z." We only use "public" and "private" as privacy labels, even though some networks allow more levels such as "viewable by friends," or even friends of friends.

The notion of what should be considered private varies from network to network and even from individual to individual within the network. To keep our model independent of the semantics of a particular network, we treat the *privacy policy* as a syntactic, exogenous labeling that specifies for every node attribute, edge, and edge attribute whether it should be public or private. Formally, it is a function $\mathsf{PP}\colon \mathcal{X} \cup \mathcal{Y} \times E \to \{\mathsf{pub}, \mathsf{priv}\}$.

We take an "operational" approach by focusing solely on node re-identification. First, it is unclear how to give a meaningful definition of social-network privacy that does not make some assumptions about the attacker's strategy and yet yields meaningful results on real-world data. Second, all currently known privacy-breaching and privacy-protection algorithms focus on node re-identification. Even edge inference, in order to be considered a meaningful privacy breach, must include learning some identifying information about the endpoints and thus implies node re-identification. Third, while anonymity is by no means sufficient for privacy[3], it is clearly necessary. A

[3]For example, suppose that the attacker can map a node in V_{aux} to a small set of nodes

re-identification algorithm that breaks anonymity is thus guaranteed to violate any reasonable definition of privacy, as long as there are any sensitive attributes at all attached to the nodes, since the algorithm re-labels the sensitive attributes with identifying information.

We define *ground truth* to be a mapping μ_G between the nodes V_{aux} of the attacker's auxiliary network and the nodes V_{san} of the target network. Intuitively, a pair of nodes are mapped to each other if they belong to the same "entity" (see Section 7.4.2). If $\mu_G(v)$ takes the special value $\perp$, then there is no mapping for node v (*e.g.*, if v was not released as part of V_{san}). Further, μ_G need not map every node in V_{san}. This is important because the overlap between V_{san} and V_{aux} may be relatively small. We do assume that the mapping is 1-1, *i.e.*, an entity has at most one node in each network, as discussed in Section 7.4.2.

Node re-identification or re-labeling refers to finding a mapping μ between a node in V_{aux} and a node in V_{san}. Intuitively, G_{aux} is a labeled graph and G_{san} is unlabeled. Node re-identification succeeds on a node $v_{\mathsf{aux}} \in V_{\mathsf{aux}}$ if $\mu(v) = \mu_G(v)$, and fails otherwise. The latter includes the case that $\mu(v) = \perp$, $\mu_G(v) \neq \perp$ and vice versa. Informally, re-identification is recognizing correctly that a given node in the anonymized network belongs to the same entity as a node in the attacker's auxiliary network.

Definition 8 (Re-identification algorithm) *A node re-identification algorithm takes as input S_{san} and S_{aux} and produces a probabilistic mapping $\tilde{\mu}\colon V_{san} \times (V_{aux} \cup \{\perp\}) \rightarrow [0,1]$, where $\tilde{\mu}(v_{aux}, v_{san})$ is the probability that v_{aux} maps to*

in V_{san} which all have the same value for some sensitive attribute. Anonymity is preserved (he does not know which of the nodes corresponds to the target node), yet he still learns the value of his target's sensitive attribute.

v_{san}.

We give such an algorithm in Section 7.5. Observe that the algorithm outputs, for each node in V_{aux}, a set of candidate nodes in V_{san} and a probability distribution over those nodes reflecting the attacker's imperfect knowledge of the re-identification mapping.

We now define the class of adversaries who attempt to breach privacy via re-identification. After constructing the mapping, the adversary updates his knowledge of the attributes of S_{aux} using the attribute values in S_{san}. Specifically, he can use the probability distribution over the candidate nodes to derive a distribution over the attribute values associated with these nodes. His success is measured by the precision of his posterior knowledge of the attributes.

Definition 9 (Mapping adversary) *A mapping adversary corresponding to a probabilistic mapping $\tilde{\mu}$ outputs a probability distribution calculated as follows:*

$$\mathsf{Adv}[X, v_{aux}, x] = \frac{\sum_{v \in V_{san}, X[v]=x} \mu(v_{aux}, v)}{\sum_{v \in V_{san}, X[v] \neq \perp} \mu(v_{aux}, v)}$$

$$\mathsf{Adv}[Y, u_{aux}, v_{aux}, y] = \frac{\sum_{u,v \in V_{san}, Y[u,v]=y} \tilde{\mu}(u_{aux}, u)\tilde{\mu}(v_{aux}, v)}{\sum_{u,v \in V_{san}, Y[u,v] \neq \perp} \tilde{\mu}(u_{aux}, u)\tilde{\mu}(v_{aux}, v)}$$

Because the auxiliary graph need not be a subgraph of the target graph, the mapping may not be complete, and the mapping adversary's posterior knowledge Adv of an attribute value is only defined for nodes v_{aux} that have actually been mapped to nodes in the target graph, at least one of which has a non-null value for this attribute. Formally, Adv is defined if there is a non-

zero number of nodes $v \in V_{\mathsf{san}}$ such that $\tilde{\mu}(v_{\mathsf{aux}}, v) > 0$ and $X[v] \neq \perp$. Edge attributes are treated similarly.

The probability of a given node having a particular attribute value can be computed in other ways, *e.g.*, by looking only at the most likely mapping. This does not make a significant difference in practice.

We say that privacy of v_{san} is compromised if, for some attribute X which takes value x in S_{san} and is designated as "private" by the privacy policy, the adversary's belief that $X[v_{\mathsf{aux}}] = x$ increases by more than δ, which is a pre-specified privacy parameter. For simplicity, we assume that the privacy policy PP is global, *i.e.*, the attribute is either public, or private for all nodes (respectively, edges).

Definition 10 (Privacy breach) *For nodes u_{aux}, $v_{aux} \in V_{aux}$, let $\mu_G(u_{aux}) = u_{san}$ and $\mu_G(v_{aux}) = v_{san}$. We say that the privacy of v_{san} is breached w.r.t. adversary Adv and privacy parameter δ if*
(a) for some attribute X such that $\mathsf{PP}[X] = \mathsf{priv}$, $\mathsf{Adv}[X, v_{aux}, x] - \mathsf{Aux}[X, v_{aux}, x] > \delta$ where $x = X[v_{aux}]$, or
(b) for some attribute Y such that $\mathsf{PP}[Y] = \mathsf{priv}$, $\mathsf{Adv}[Y, u_{aux}, v_{aux}, y] - \mathsf{Aux}[Y, u_{aux}, v_{aux}, y] > \delta$ where $y = Y[u_{aux}, v_{aux}]$.

Definition 10 should be viewed as a meta-definition or a template, and must be carefully adapted to each instance of the re-identification attack and each concrete attribute. This involves subjective judgment. For example, did a privacy breach occur if the the attacker's confidence increased for some attributes and decreased for others? Learning common-sense knowledge from the sanitized network (for example, that all nodes have fewer than 1000 neighbors) does not intuitively constitute a privacy breach, even though it satisfies

Definition 10 for the "node degree" attribute. Such common-sense knowledge must be included in the attacker's Aux. Then learning it from the sanitized graph does not constitute a privacy breach.

7.4.6 Measuring success of an attack

While it is tempting to quantify de-anonymization of social networks in terms of the fraction of nodes affected, this results in a fairly meaningless metric. Consider the following thought experiment. Given a network $G = (V, E)$, imagine the network G' consisting of G augmented with $|V|$ singleton nodes. Re-identification fails on the singletons because there is no edge information associated with them, and, therefore, the naïve metric returns half the value on G' as it does on G. Intuitively, however, the presence of singletons should not affect the performance of any de-anonymization algorithm.

This is not merely hypothetical. In many online networks, the majority of nodes show little or no observable activity after account creation. Restricting one's attention to the giant connected component does not solve the problem, either, because extraneous nodes with degree 1 instead of 0 would have essentially the same (false) impact on naïvely measured performance.

Instead, we assign a weight to each affected node in proportion to its importance in the network. Importance is a subjective notion, but can be approximated by node *centrality*, which is a well-studied concept in sociology that only recently came to the attention of computer scientists [135, 68, 175, 15, 152].

There are three groups of centrality measures: local, eigenvalue-based and distance-based. Local methods such as degree centrality consider only

the neighbors of the node. Eigenvalue methods also consider the centrality of each neighbor, resulting in a convergent recursive computation. Distance-based measures consider path lengths from a node to different points in the network. A well-known eigenvalue-based measure was proposed by Bonacich in [37], while [128] presents a textbook treatment of centrality.

We find that the decision to use a centrality measure at all, as opposed to a naïve metric such as the raw fraction of nodes de-anonymized, is much more important than the actual choice of the measure. We therefore use the simplest possible measure, degree centrality, where each node is weighted in proportion to its degree. In a directed graph, we use the sum of in-degree and out-degree.

There is an additional methodological issue. For a mapped pair of nodes, should we use the centrality score from the target graph or the auxiliary graph? It is helpful to go back to the pathological example that we used to demonstrate the inadequacy of fraction-based metrics. If either of the nodes in the mapped pair is a singleton, then the de-anonymization algorithm clearly has no hope of finding that pair. Therefore, we compute the centrality in both graphs and take the minimum of the two. We believe that this formulation captures most closely the spirit of the main question we are answering in this work: "what proportion of entities that are active in a social network and for which non-trivial auxiliary information is available can be re-identified?"

Given a probabilistic mapping $\tilde{\mu}$, we say that a (concrete) mapping is *sampled* from $\tilde{\mu}$ if for each u, $\mu(u)$ is sampled according to $\tilde{\mu}(u, .)$.

Definition 11 (Success of de-anonymization) *Let* $V_{\mathsf{mapped}} = \{v \in V_{\mathsf{aux}} : \mu_G(v) \neq \perp\}$. *The* success rate *of a de-anonymization algorithm outputting a*

probabilistic mapping $\tilde{\mu}$, w.r.t. a centrality measure ν, is the probability that μ sampled from $\tilde{\mu}$ maps a node v to $\mu_G(v)$ if v is selected according to ν:

$$\frac{\sum_{v \in V_{mapped}} \mathbf{P}[\mu(v) = \mu_G(v)]\nu(v)}{\sum_{v \in V_{mapped}} \nu(v)}$$

The error rate *is the probability that μ maps a node v to any node other than $\mu_G(v)$:*

$$\frac{\sum_{v \in V_{mapped}} \mathbf{P}[\mu(v) \neq \perp \wedge \mu(v) \neq \mu_G(v)]\nu(v)}{\sum_{v \in V_{mapped}} \nu(v)}$$

The probability is taken over the inherent randomness of the de-anonymization algorithm as well as the sampling of μ from $\tilde{\mu}$. Note that the error rate includes the possibility that $\mu_G(v) = \perp$ and $\mu(v) \neq \perp$.

The above measure only gives a lower bound on privacy breach because privacy can be violated without complete de-anonymization. Therefore, if the goal is to protect privacy, it is *not* enough to show that this measure is low. It is also necessary to show that Definition 10 is not satisfied. Observe, for example, that simply creating k copies of the graph technically prevents de-anonymization and even satisfies naïve syntactic definitions such as k-anonymity, while completely violating any reasonable definition of privacy.

In the other direction, however, breaking Definition 11 for a large fraction of nodes—as our algorithm of Section 7.5 does—is sufficient to break privacy via Definition 10, as long some trivial conditions are met: at least one private attribute is released as part of $\mathcal{X}_{\mathsf{san}}$, and the adversary possesses little or no auxiliary information about this attribute.

7.5 De-anonymization

Our re-identification algorithm runs in two stages. First, the attacker identifies a small number of "seed" nodes which are present both in the anonymous target graph and the attacker's auxiliary graph, and maps them to each other. The main, propagation stage is a self-reinforcing process in which the seed mapping is extended to new nodes using only the topology of the network, and the new mapping is fed back to the algorithm. The eventual result is a large mapping between subgraphs of the auxiliary and target networks which re-identifies all mapped nodes in the latter.

7.5.1 Seed identification

While algorithms for seed identification are not our primary technical contribution, they are a key step in enabling our overall algorithm to succeed. Here we describe one possible seed identification algorithm. The attacks in [23] can also be considered seed identification algorithms. We briefly discuss alternatives at the end of Section 7.6.1.

We assume that the attacker's individual auxiliary information (see Section 7.4.4) consists of a clique of k nodes which are present both in the auxiliary and the target graphs. It is sufficient to know the degree of each of these nodes and the number of common neighbors for each pair of nodes.

The seed-finding algorithm takes as inputs (1) the target graph, (2) k seed nodes in the auxiliary graph, (3) k node-degree values, (4) $\binom{k}{2}$ pairs of common-neighbor counts, and (5) error parameter ϵ. The algorithm searches the target graph for a unique k-clique with matching (within a factor of $1 \pm \epsilon$) node degrees and common-neighbor counts. If found, the algorithm maps

the nodes in the clique to the corresponding nodes in the auxiliary graph; otherwise, failure is reported.

While this brute-force search is exponential in k, in practice this turns out not to be a problem. First, if the degree is bounded by d, then the complexity is $O(nd^{k-1})$. Second, the running time is heavily input-dependent, and the inputs with high running time turn out to produce a large number of matches. Terminating the algorithm as soon as more than one match is found greatly decreases the running time.

7.5.2 Propagation

The propagation algorithm takes as input two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ and a partial "seed" mapping μ_S between the two. It outputs a mapping μ. One may consider probabilistic mappings, but we found it simpler to focus on deterministic 1-1 mappings $\mu \colon V_1 \to V_2$.

Intuitively, the algorithm finds new mappings using the topological structure of the network and the feedback from previously constructed mappings. It is robust to mild modifications of the topology such as those introduced by sanitization. At each iteration, the algorithm starts with the accumulated list of mapped pairs between V_1 and V_2. It picks an arbitrary unmapped node u in V_1 and computes a score for each unmapped node v in V_2, equal to the number of neighbors of u that have been mapped to neighbors of v. If the strength of the match (see below) is above a threshold, the mapping between u and v is added to the list, and the next iteration starts. There are a few additional details and heuristics that we describe below.

Eccentricity. Eccentricity is a heuristic defined in [194] in the context

of de-anonymizing databases. It measures how much an item in a set X "stands out" from the rest, and is defined as

$$\frac{\mathsf{max}(X) - \mathsf{max}_2(X)}{\sigma(X)}$$

where max and max_2 denote the highest and second highest values, respectively, and σ denotes the standard deviation.

Our algorithm measures the eccentricity of the set of mapping scores (between a single node in v_1 and each unmapped node in v_2) and rejects the match if the eccentricity score is below a threshold.

Edge directionality. Recall that we are dealing with directed graphs. To compute the mapping score between a pair of nodes u and v, the algorithm computes two scores–the first based only on the incoming edges of u and v, and the second based only on the outgoing edges. These scores are then summed.

Node degrees. The mapping scores as described above are biased in favor of nodes with high degrees. To compensate for this bias, the score of each node is divided by the square root of its degree. The resemblance to cosine similarity[4] is not superficial: the rationale is the same.

Revisiting nodes. At the early stages of the algorithm, there are few mappings to work with, and therefore the algorithm makes more errors. As the algorithm progresses, the number of mapped nodes increases and the error rate goes down. Thus the need to revisit already mapped nodes: the mapping computed when revisiting a node may be different because of the new mappings that have become available.

[4]The *cosine similarity measure* between two sets X and Y is defined when neither is empty: $\mathsf{cos}(X,Y) = \frac{|X \cap Y|}{\sqrt{|X||Y|}}$.

Reverse match. The algorithm is completely agnostic about the semantics of the two graphs. It does not matter whether G_1 is the target graph and G_2 is the auxiliary graph, or vice versa. Each time a node u maps to v, the mapping scores are computed with the input graphs switched. If v gets mapped back to u, the mapping is retained; otherwise, it is rejected.

The following pseudocode describes the algorithm in detail. `theta` is a parameter that controls the tradeoff between the yield and the accuracy.

```
function propagationStep(lgraph, rgraph, mapping)

  for lnode in lgraph.nodes:
    scores[lnode] = matchScores(lgraph, rgraph, mapping, lnode)
    if eccentricity(scores[lnode]) < theta: continue
    rnode = (pick node from rgraph.nodes where
          scores[lnode][node] = max(scores[lnode]))

    scores[rnode] = matchScores(rgraph, lgraph, invert(mapping), rnode)
    if eccentricity(scores[rnode]) < theta: continue
    reverse_match = (pick node from lgraph.nodes where
          scores[rnode][node] = max(scores[rnode]))
    if reverse_match != lnode:
      continue

    mapping[lnode] = rnode

function matchScores(lgraph, rgraph, mapping, lnode)
```

```
  initialize scores = [0 for rnode in rgraph.nodes]

  for (lnbr, lnode) in lgraph.edges:
    if lnbr not in mapping: continue
    rnbr = mapping[lnbr]
    for (rnbr, rnode) in rgraph.edges:
      if rnode in mapping.image: continue
      scores[rnode] += 1 / rnode.in_degree ^ 0.5

  for (lnode, lnbr) in lgraph.edges:
    if lnbr not in mapping: continue
    rnbr = mapping[lnbr]
    for (rnode, rnbr) in rgraph.edges:
      if rnode in mapping.image: continue
      scores[rnode] += 1 / rnode.out_degree ^ 0.5

  return scores

function eccentricity(items)

  return (max(items) - max2(items)) / std_dev(items)

until convergence do:
  propagationStep(lgraph, rgraph, seed_mapping)
```

Complexity. Ignoring revisiting nodes and reverse matches, the complexity of the algorithm is $O(|E_1|d_2)$, where d_2 is a bound on the degree of the nodes in V_2. To see this, let μ_{part} be the partial mapping computed at any stage of the algorithm. For each $u \in V_1$ and each v adjacent to u such that $v \in \mathsf{domain}(\mu_{part})$, the algorithm examines each of the neighbors of $\mu_{part}(v)$, giving an upper bound of $|E_1|d_2$.

Assuming that a node is revisited only if the number of already-mapped neighbors of the node has increased by at least 1, we get a bound of $O(|E_1|d_1d_2)$, where d_1 is a bound on the degree of the nodes in V_1. Finally, taking reverse mappings into account, we get $O((|E_1| + |E_2|)d_1d_2)$.

7.6 Experiments

We used data from three large online social networks in our experiments. The first graph is the "follow" relationships on the Twitter microblogging service, which we crawled in late 2007. The second graph is the "contact" relationships on Flickr, a photo-sharing service, which we crawled in late 2007/early 2008. Both services have APIs that expose a mandatory username field, and optional fields name and location. The latter is represented as free-form text. The final graph is the "friend" relationships on the LiveJournal blogging service; we obtained it from the authors of [184].

Typically, a network crawl can only recover the giant connected component. Both Twitter and Flickr allow to query only forward links. Therefore, we can expect to recover the strongly-connected component (SCC) fully and the weakly connected component (WCC) incompletely.

We crawled the entire SCC of Twitter, subject to the caveat that the

Table 7.1: Graphs used

Network	Nodes	Edges	Av. Deg
Twitter	224K	8.5M	37.7
Flickr	3.3M	53M	32.2
LiveJournal	5.3M	77M	29.3

Twitter API for discovering relationships is indirect; in particular, we cannot discover users whose activity on the website is "protected," *i.e.*, viewable by friends only. Interestingly, the size of the Twitter user population, at least as reflected in the connected component of regular users, turned out to be much smaller than was being reported in the media at the time of our crawl. It is also worth noting that since then Twitter has introduced crippling rate limitations on its API, which make a large-scale crawl infeasible.

We could not crawl the entire SCC of the Flickr graph due to its size. We crawled it in a priority-queue fashion, giving the highest priority to the nodes with the highest number of incoming edges from the already crawled nodes. Comparing our numbers with [184], we conclude that we have, in fact, recovered most of the SCC.

Finally, the authors of [184], who kindly provided with us with the LiveJournal data, report that their crawl covers the vast majority of the users in LiveJournal's WCC.

The parameters of the three graphs are summarized in Table 7.1. In computing the average degree, the degree of a node is counted as the sum of its in- and out-degrees.

7.6.1 Seed identification

To demonstrate feasibility of seed identification, we ran the algorithm of Section 7.5.1 with the LiveJournal graph as its target. Recall from Section 7.4.4 that the auxiliary information needed to create seed mappings comes from the users of the target network. Therefore, we can evaluate feasibility of seed identification simply by measuring how much auxiliary information is needed to identify a unique node in the target graph. We emphasize that our main de-anonymization algorithm needs only a handful of such nodes.

For simplicity, we assume that the attacker only has access to the undirected graph, where an edge is included only if it is symmetrical in the original graph. This *underestimates* the re-identification rate, because the attacker would have more information if directionality of edges were considered.

We synthetically generate auxiliary information for seed identification starting from randomly sampled *cliques*. To sample a clique of size k, we start from a random node and, at every stage, randomly pick a node which is adjacent to all the nodes picked so far. If there is no such node, we start over.

This method does not sample uniformly from all the cliques in the graph; the distribution of selected nodes is much more equitable. If we sample a k-clique uniformly, it is susceptible to anomalies in the graph that make the result meaningless. If the graph has a large clique, or even a large dense subgraph, then almost every k-clique sampled will belong to this large clique or subgraph.

Given a clique (specifically, a 4-clique), we assume that the attacker knows the degrees of these 4 nodes as well as the number of common neighbors of each of the 6 pairs. The auxiliary information may be imprecise, and the

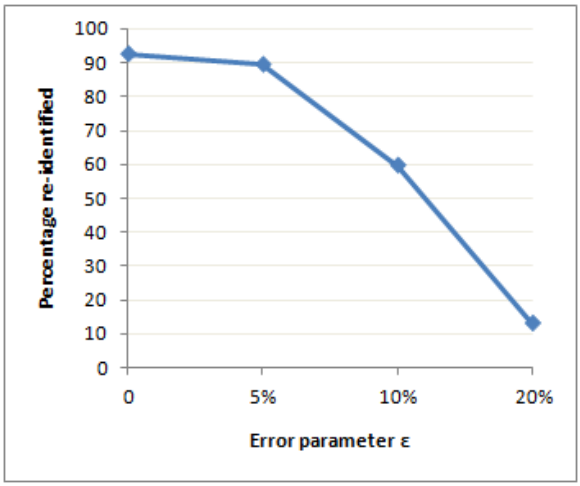

Figure 7.1: Seed identification

search algorithm treats a 4-clique in the target graph as a match as long as each degree and common-neighbor count matches within a factor of $1 \pm \epsilon$, where ϵ is the error parameter (intuitively, the higher the error, the noisier the auxiliary information and the lower the re-identification rate). Figure 7.1 shows how re-identification rate decreases with noise. Recall that we allow at most one match, and so the attacker never makes an error as long as his assumptions about the imprecision of his auxiliary information are correct.

This experiment establishes that seed identification is feasible in practice. If anything, it underestimates how easy this is to do in the real world, where the attacker can use auxiliary information other than degrees and common-neighbor counts. Searching based on the structure of the target users' graph neighborhoods allows re-identification with just two or even a single node, although this is algorithmically complex.

7.6.2 Propagation

Robustness against perturbation and seed selection

The most remarkable feature of our propagation algorithm is that it achieves "viral," self-reinforcing, large-scale re-identification regardless of the number of seeds, as long as the latter is above a (low) threshold. To study this behavior, we carried out an experiments on pairs of subgraphs, over 100,000 nodes each, of a real-world social network. In each experiment, one of the subgraphs was used as the auxiliary information, the other as the target. The graphs were artificially perturbed by adding different levels of noise to achieve various degrees of edge overlap.

Perturbation strategy. Given a real network graph $G = (V, E)$, our goal is to sample subsets V_1, V_2 of V such that V_1 and V_2 have an overlap of α_V. Overlap is measured in terms of the Jaccard Coefficient, which is defined for two sets X and Y if one of them is non-empty: $JC(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. Thus, if each of two sets shares half its members with the other, the overlap is $\frac{1}{3}$. We simply partition V randomly into three subsets V_A, V_B, V_C of size $\frac{1-\alpha_V}{2}|V|, \alpha_V|V|, \frac{1-\alpha_V}{2}|V|$, respectively, and set $V_1 = V_A \cup V_B$ and $V_2 = V_B \cup V_C$.

We use one subgraph as the auxiliary information and the other as the anonymous target graph. As mentioned in Section 7.2, we believe that introducing noise via edge deletions and additions is the only realistic method of perturbing the edges. Our goal is to simulate the effect of perturbation on the target graph as follows (Procedure A):

- Derive E' from E by adding edges.
- Derive E'' from E' by randomly deleting edges.

- Project E and E'' on V_1 and V_2, respectively, to obtain E_1 and E_2.

The best way to add edges is to use link prediction, which will result in plausible fake edges. Instead of choosing a specific link prediction algorithm, we perform the following (Procedure B):

- Make two copies of E and independently delete edges at random from each copy.
- Project the copies on V_1 and V_2, respectively, to get E_1 and E_2.

It should be clear that Procedure B produces more plausible edges than even the best concrete link prediction algorithm. If the link prediction algorithm is *perfect*, *i.e.*, if the edge additions accomplish the reverse of random edge deletion, then the two procedures are more or less equivalent (E' in Procedure A corresponds to E in Procedure B; E and E'' in Procedure A correspond to the two perturbed copies in Procedure B). If the link prediction is not perfect, then Procedure B is better in the sense that it leads to more realistic noise, and thus makes the task of our de-anonymization algorithm harder.

This leaves the question of what fraction β of edges to remove to get an edge overlap of α_E. The fraction of common edges is $(1-\beta)^2$, while the fraction of edges left in at least one of the copies is $1-\beta^2$, giving $\frac{(1-\beta)^2}{1-\beta^2} = \alpha_E$, which yields $\beta = \frac{1-\alpha_E}{1+\alpha_E}$ as the only valid solution. Note that the edge overlap is calculated for the subgraphs formed by the overlapping nodes. The overlap between E_1 and E_2 is much lower.

Results. We investigated the impact that the number of seeds has on the ability of the propagation algorithm to achieve large-scale re-identification, and also its robustness to perturbation.

Figure 7.2 shows that the selection of seeds determines whether propagation step dies out or not (cf. phase transition [257]), but whenever large-scale propagation has been achieved, the re-identification rate stays remarkably constant. We find that when the algorithm dies out, it re-identifies no more than a few dozen nodes correctly.

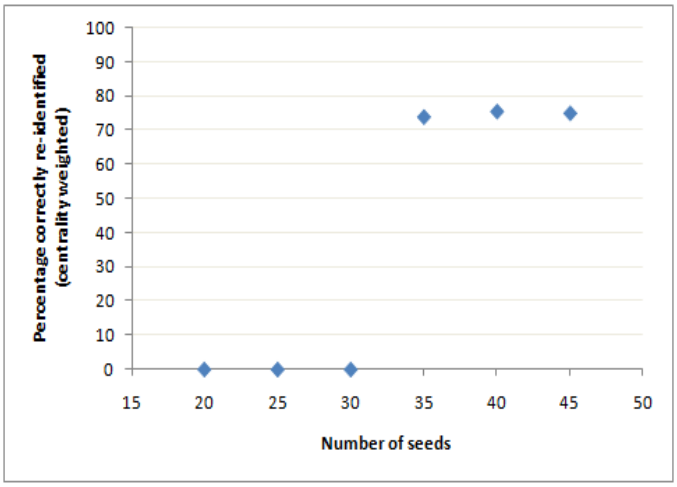

Figure 7.2: The fraction of nodes re-identified depends sharply on the number of seeds. Node overlap: 25%; Edge overlap: 50%

We performed a further experiment to study the phase transition better. A run is classified as successful if it re-identifies at least 1,000 nodes. Figure 7.3 shows the resulting probabilities of large-scale propagation. The phase transition is somewhat less sharp than might appear from Figure 7.2, although the window is almost completely in the range [15,45].

It must be noted that the number of seeds required to trigger propagation depends heavily on the parameters of the graph and the algorithm used for seed selection. We therefore caution against reading too much into the numbers. What this experiment shows is that a phase transition does happen and that it is strongly dependent on the number of seeds. Therefore, the ad-

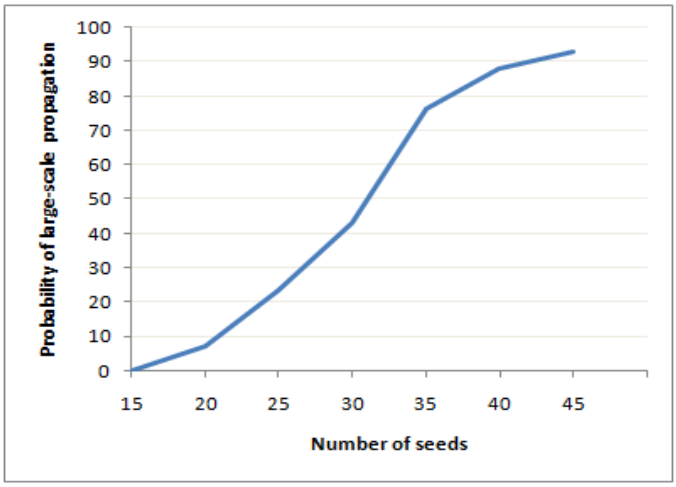

Figure 7.3: The phase transition in more detail. Node overlap: 25%; Edge overlap: 50%

versary can collect seed mappings incrementally until he has enough mappings to carry out large-scale re-identification.

Figure 7.4 shows that imprecision of the auxiliary information decreases the percentage of nodes re-identified, but cannot prevent large-scale re-identification.

Mapping between two real-world social networks

As our main experiment, we ran our propagation algorithm with the graph of Flickr as the auxiliary information and the anonymous graph of Twitter as the target.

Ground truth. To verify our results, we had to determine the *ground truth*, *i.e.*, the true mapping between the two graphs. We produced ground-truth mappings based on exact matches in either the username, or name field. Once a match is found, we compute a score based on a variety of heuristics on all three fields (username, name and location). If the score is too low, we reject

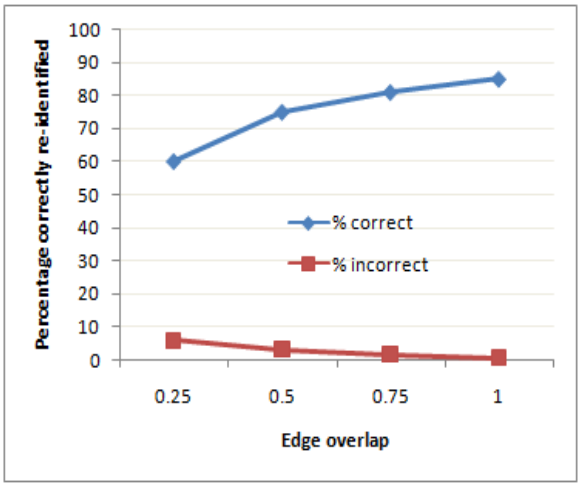

Figure 7.4: Effect of noise. Node overlap: 25%; Number of seeds: 50

the match as spurious.

- For usernames, we use the length to measure the likelihood that a username match is spurious. The rationale is that a username such as "tamedfalcon213" is more likely to be identifying than "joe".
- For names, we use the length of the names, as well as the frequency of occurrence of the first and last names. Rarer names indicate a stronger match.
- For locations, we use heuristics such as two-letter state abbreviations.

This resulted in around 27,000 mappings, which we will call $\mu(G)$. Since these mappings were computed with a completely different information than used by the de-anonymization algorithm, errors in the ground truth can only degrade the reported performance of our de-anonymization algorithm. We picked a random sample of the mappings and verified by human inspection that the error rate is well under 5%.

Of course, some of those who use both Flickr and Twitter may use completely different usernames and names on the two services and are thus not included in our ground-truth mappings. This has no effect on the reported performance of our algorithm. When it does recognize two nodes as belonging to the same user, it is rarely wrong, and, furthermore, it can successfully re-identify thousands of users.

It is possible that our algorithm has a better performance on the nodes where the ground truth is known than on other nodes. For example, users who acquire distinctive usernames on both websites might be habitual early adopters of web services. Thus, the numbers below must be interpreted with caution.

Our seed mapping consisted of 150 pairs of nodes selected randomly from $\mu(G)$, with the constraint that the degree of each mapped node in the auxiliary graph is at least 80. More opportunistic seed selection can lower the number of seeds required.

The accuracy of our algorithm on $\mu(G)$ (weighted by centrality—see Section 7.4.6) is summarized below:

- 30.8% of the mappings were re-identified correctly, 12.1% were identified incorrectly, and 57% were not identified.

- 41% of the incorrectly identified mappings (5% overall) were mapped to nodes which are at a distance 1 from the true mapping. It appears likely that human intelligence can be used to complete the de-anonymization in many of these cases.

- 55% of the incorrectly identified mappings (6.7% overall) were mapped

to nodes where the same geographic location was reported.[5] Thus, even when re-identification does not succeed, the algorithm can often identify a node as belonging to a cluster of similar nodes, which might reveal sensitive information (recall the discussion in Section 7.4.5).

- The above two categories overlap; of all the incorrect mappings, only 27% (or 3.3% overall) fall into neither category and are completely erroneous.

7.7 Summary

The main lesson of this chapter is that anonymity is not sufficient for privacy when dealing with social networks. We developed a generic re-identification algorithm and showed that it can successfully de-anonymize several thousand users in the anonymous graph of a popular microblogging service (Twitter), using a completely different social network (Flickr) as the source of auxiliary information.

Our experiments underestimate the extent of the privacy risks of anonymized social networks. The overlap between Twitter and Flickr membership at the time of our data collection was relatively small. Considering only the users who supplied their names (about a third in either network), 24% of the names associated with Twitter accounts occur in Flickr, while 5% of the names associated with Flickr accounts occur in Twitter. Since human names are not unique, this overestimates the overlap in membership. By contrast, 64% of Facebook users are also present on MySpace [208]. As social networks grow

[5]This was measured by sampling 200 of the erroneous mappings and using human analysis. We consider the geographical location to be the same if it is either the same non-U.S. country, or the same U.S. state.

larger and include a greater fraction of the population along with their relationships, the overlap increases. Therefore, we expect that our algorithm can achieve an even greater re-identification rate on larger networks.

We demonstrated feasibility of successful re-identification based solely on the network topology and assuming that the target graph is completely anonymized. In reality, anonymized graphs are usually released with at least some attributes in their nodes and edges, making de-anonymization even easier. Furthermore, any of the thousands of third-party application developers for popular online social networks, the dozens of advertising companies, governments who have access to telephone call logs, and anyone who can compile aggregated graphs of the form described in Section 7.2 have access to auxiliary information which is much richer than what we used in our experiments. At the same time, an ever growing number of third parties get access to sensitive social-network data in anonymized form. These two trends appear to be headed for a collision resulting in major privacy breaches, and any potential solution would appear to necessitate a fundamental shift in business models and practices and clearer privacy laws on the subject of Personally Identifiable Information (see Section 7.3.4).

Chapter 8

Privacy Risks of Collaborative Filtering

8.1 Introduction

Recommender systems have become ubiquitous on major e-commerce sites. When you buy products from Amazon, rent movies on Netflix, find friends on Facebook, or perform myriad other tasks online, Web-based recommender systems suggest options to you based on your choices and behavior. These suggestions are often a product of patterns learned from past customers: for example, the system may inform you that users who bought item X (as you just did) often buy item Y.[1] Your own purchases and/or ratings are used to generate recommendations for other customers and to refine the system's model of your own behavior. Such recommender systems are particularly useful

[1]For clarity, we often use terminology appropriate for an online retailer, like "customers" and "purchases," throughout this chapter. Our analysis is equally applicable, however, to other sites employing recommender systems.

for spotting both expected and unexpected trends quickly and automatically, even given a massive quantity of real-time transaction data. This can benefit both retailers and consumers. Section 8.2 describes recommender systems in greater detail.

Our objective in this work is to investigate the privacy risk of recommender systems, *i.e.*, the risk that the output of the recommendation algorithm may expose sensitive details of an individual user's behavior without his or her knowledge or consent.

At first glance, this risk does not appear significant. Datasets underlying typical recommender systems on popular websites are large and elaborate, involving tens of thousands to millions of users with dozens of actions each. Without access to the dataset (which is usually *not* revealed to the customers of the service), making inferences about transactions of specific users does not appear feasible. Furthermore, in many cases the recommendations are only indirectly related to the behavior of individual users. For example, recommendations on Amazon.com and many other online retailers (Section 8.2.1) are computed based on similarities between pairs of items rather than pairs of users. While item similarity lists are shown to the customer, these merely reveal items whose purchases tend to be correlated, without linking these items to specific users who purchased them.

Our first contribution is to develop a set of practical algorithms that allow accurate inference of (partial) individual behavior from the aggregate output of a typical recommender system. As a warm-up exercise, we present a simple active attack, in which the attacker creates sybil users whose behavior is similar to what the attacker knows about the target user's behavior. The attacker infers the target's hidden actions using the recommendations made

by the system to these sybils.

Our main attack is purely *passive*. The attacker need not create fake users, make any purchases or enter ratings into the system. He passively watches the item similarity lists produced by the system, and uses the observed changes in the items' relative rank, along with his knowledge of some of the target user's purchases (acquired from public ratings, blog posts, and so on), to infer other purchases made by this user. The items we purchase, news stories we read, videos we watch, and other choices that we make reveal details about who we are. These details may be as innocuous as our favorite sports teams or as sensitive as health concerns. The fact that they can be unintentionally revealed by a recommender system presents a significant risk to individual privacy.

This attack demonstrates that recommender systems that update their suggestions frequently can reveal user behavior at a fine level of granularity. Our algorithms exploit the fact that online retailers tend to offer vast catalogs of items, including a large number of less popular items that are purchased relatively infrequently. A purchase of such an item is likely to result in observable changes in the item similarity lists, allowing the attacker to make accurate inferences. It is worth mentioning that the abundance of relatively unpopular items in online catalogs, sometimes referred to as the "Long Tail" phenomenon [16], was previously exploited to de-anonymize movie-rating records of the Netflix Prize dataset [194]. Our focus in this work is very different: instead of de-anonymizing a dataset, we aim to infer its contents from the output of a sophisticated recommender system.

Our second contribution is to evaluate our techniques on a large, real-world recommender system used by Amazon. With the same access to the

Amazon website as available to any customer, an attacker can use our method to make predictions about purchases of individual customers. In Section 8.5, we perform a limited analysis of Amazon's recommendations and consider several Amazon customers that appear to purchase products from Amazon with some regularity, as reflected by their public reviews on the Amazon website. When our algorithm observed that another item entered or climbed in the list of top 10 most similar items for several products already purchased by the customer, it inferred that the customer likely purchased the item. For the customers discussed, freely available sales rank data helped strengthen our belief that the predicted purchases occurred. Our predictions were validated when the customers posted public reviews of the predicted items at a later date. Based on simulating our technique on the Netflix Prize dataset (see below), we expect that our predictions have a fairly high accuracy: for example, predictions supported by 9 related items are expected to have 50% accuracy. Below, we explain how this accuracy was estimated; short of cooperating with Amazon, however, there is no general method for verifying our predictions of Amazon purchases which have not been voluntarily revealed by the user. This presents an interesting methodological challenge.

Our third contribution is to develop a method for measuring the accuracy of recommendation-based inferences. We re-implemented an Amazon-like recommender system from the publicly available details. Instead of applying it to the actual Amazon transactions (to which we do not have access), we applied it to the publicly available Netflix Prize dataset; the differences between the datasets are discussed in Section 8.4. The Netflix dataset contains dated movie ratings of 500,000 subscribers of the Netflix DVD rental service [196]. It serves as the "ground-truth oracle" for verifying predictions made by our

algorithm. Whenever it infers, given the output of a recommender system (but not the dataset itself!), that customer X must have rated movie Y on date D, we can verify the inference by consulting the customer's actual transactions in the dataset. It also allows us to evaluate the tradeoff between the number of predictions and their accuracy. For example, with 10,000 customers, each of whom has publicly rated 100 items, the algorithm makes around 500 predictions per month, of which 200 are correct.

We believe this work is the first to develop a generic passive method for inferring individual customer information from the output of a recommender system, and to empirically analyze and quantify privacy risks of real-world, deployed systems.

8.2 Recommender systems

Recommender systems have become a vital business tool in attracting and keeping users on e-commerce websites. The business case for recommender systems is presented in [107]. Virtually every major Web destination where users purchase, rent, or view items out of a catalog has a feature to recommend new items to returning users based on their past behavior. This includes the market leaders in books (Amazon), music (pandora, last.fm), movies (Netflix), videos (YouTube), news aggregation (Google News) and blogs (Google Reader).

A technical survey of the literature can be found in Adomavicius and Tuzhilin [9]. The task of a recommender system can be abstractly described as follows. There is a matrix whose rows are users and whose columns are items; this matrix represents transactions: purchases, ratings, *etc.*, depending on the

application. Furthermore, there is some "profile" information associated with each user (such as demographics) and "metadata" associated with each item (such as price and category). The matrix is incomplete, and in fact "sparse:" only a small fraction of entries are filled in. The goal of a recommender system is to *extrapolate* the unknown entries of the matrix in a way that most closely matches their "true" values.

Recommender systems can be classified as content-based, collaborative, and hybrid. Content-based systems work by identifying relationships between items based on metadata and recommending items that are similar to the user's past transactions. Collaborative recommendation is much more robust and domain-agnostic, and hence very popular; it works by identifying relationships between items based on the ratings of other users, ignoring metadata. Hybrid systems use a mix of the two techniques.

Content-based recommender systems have no privacy drawbacks, since the system does not consider any other user's transactions in making recommendations for a given user. Collaborative systems, however, are by far the most popular variety and will be our focus of the rest of this chapter.

Traditionally, collaborative filtering methods were *user-based*. For a given user, the system finds other users who have a similar transaction history and recommends new items based on the past transactions of these users. The alternative is *item-based* collaborative filtering, which we now discuss.

8.2.1 Item-to-item recommendations and a broader definition

Item-based collaborative filtering was first described in a paper by Sarwar *et al* [224]. Linden *et al.* pioneered the use of item-based recommendations at Amazon.com. Their work is described in a U.S. patent [166] and a technical paper [165]. The key idea is to compute *similarity scores* for each pair of items based on the likelihood of the items being purchased by the same customer. There are two uses for these scores:

- To compute *item similarity lists* which are used to suggest other items related to a given item;
- To make recommendations to users by selecting items that are similar to the user's previously purchased items.

Item-based collaborative filtering has since become popular [30]. In addition, it has become the standard practice for Web destinations to provide users with item similarity lists.[2] On Amazon.com, this is seen as the "customers who bought this item also bought ..." feature. Similar features are found on most e-commerce websites, including all listed above. Item similarity lists appear on many sites that do not have traditional user-to-item recommendations, such as IMDb and news sites such as CNN and the New York Times. Even on sites that have both, such as YouTube, the item-to-item recommendation feature is much more prominent.[3]

[2] A survey of algorithms can be found in [75].

[3] Our use of the term item-to-item recommendation is slightly different from Linden *et al.* [165]. We summarize our terminology in Appendix E.

The reason for this phenomenon is that item-to-item recommendations serve a dual purpose: not only do they drive sales or rentals, they are also a navigational aid that increases user retention on the site. Of course, where the revenue model is based on monetizing user attention through advertisements, these two incentives converge.

In the offline world, such as supermarkets, item-item similarity analysis is frequently deployed in order to optimize the store layout [42]. The point-of-sale system records customer purchases, and items that are purchased together are considered similar. This is used to optimize the physical layout of the inventory by placing similar items adjacent to each other.

Finally, user-to-user recommendations can be seen on last.fm and librarything.com, where they are based on similarities in tastes and transactions, on various dating websites such as eharmony.com, where they are based on explicitly specified user preferences and match constraints, and on social networking sites such as Facebook ("people you might know") where they are based on the structure of the network itself, *i.e.*, existing user-user links.

In recognition of this business reality, we propose that recommender systems be redefined to include all three categories of recommendations: the traditional notion of user-to-item recommendations, as well as item-to-item and user-to-user recommendations. Item-to-item recommendations, in particular, are very relevant to our privacy analysis, as will become clear in the next section.

8.3 Privacy Risks in Recommender Systems

By design, recommender systems are intended to provide advice to users based on the information about other users' behavior, *i.e.*, their transactions, expressed preferences, *etc.* Therefore, in a very abstract sense, the risk of disclosing information about individuals is an inherent feature of recommender systems. This abstract risk, however, does not directly imply that the output of a recommender system may violate the privacy of a specific user. In particular, it is not clear *a priori* whether it is possible to draw meaningful conclusions about any given user whose data were used, among those from thousands of other users, to create public recommendations. Understanding and quantifying the potential for privacy breaches caused by recommender systems is the primary contribution of this work.

Previous work on privacy risks of recommender systems focused on straddlers [216], whose tastes span unrelated genres. Crucially, this work assumed a very strong attack model, in which the attacker is given the entire database of individual transaction histories (*i.e.*, the user-item transaction matrix), with all entries anonymized. This model is applicable in scenarios where the attacker has access to the raw inputs of the recommendation algorithm, *e.g.*, when the collaborative filtering functionality is outsourced, but is clearly beyond the resources of a casual attacker.

By contrast, we focus on a much more realistic setting whether the attacker has only a "black-box" view of the recommender system. In particular, he does not have privileged access to the raw inputs. Just like any other user, the attacker observes only the *outputs*, *i.e.*, public recommendations provided by the system (below, we describe several forms these recommendations may

take). Therefore, our attacks can be effectively carried out by any user of the system.

In the rest of this chapter, we demonstrate that, in combination with a small amount of auxiliary information, the output of common recommender systems can be used to accurately infer sensitive information about the actions of users that have been used as inputs into the system, but are *not* directly observable by the attacker.

Attack model and attacker's auxiliary information. Depending on the design of the recommender system and the interface it provides to the users, the following pieces of information may be public and thus available to the attacker: (1) the item similarity list, (2) the sales rank of each item, (3) for some users, a subset of the items they purchased or rated, (4) the user-item similarity score, *i.e.*, the actual recommendation made by the system when it suggests a particular item to a specific user (this is essential for active attacks, where the attacker directly controls some of the users), and (5) the user similarity list. We now survey these in more detail and explain how they can be used for attacks of different types.

The *item similarity list* usually takes the form of "users who bought this item also bought those items," or "users who liked this movie also enjoyed those movies." For example, item similarity lists are among the primary outputs of the Amazon recommendation algorithm. The items on a given list have varying similarity scores, which are almost never publicly available. The size of the similarity list depends on the system or even on the access mode: for instance, the Amazon API [22] only provides 10 similar items, while manual access to the Amazon website provides up to 100 (and perhaps more).

Sales rank of each item is often available. In the crudest form, it simply identifies the bestsellers, but in the case of Amazon, the precise sales rank of each item is available through the API.

We assume that for some users, a *subset of their transaction history* is available. This is a realistic assumption because many users disclose some of their transactions voluntarily, through the service itself. On Amazon, many users publicly rate some of the items they purchased. This auxiliary information about specific users can also be collected from blog posts, conversations with co-workers, and so on. We emphasize that the latter source of auxiliary information is completely outside the control of the recommendation service provider.

Our passive attack uses only the item similarity list, sales rank, and partial user history. Intuitively, it observes the similarity lists associated with the items that are known to have been bought or rated by a specific user. Other items purchased or rated by this user–even if not included in the observable partial history–are more likely to appear on the similarity list. Our algorithm uses changes in the item similarity lists over time (*e.g.*, appearance of an item or an increase in its rank) to infer conclusions about this "hidden" activity.

We also describe an active attack, which assumes that in addition to the partial user history, the attacker has access to the user-item similarity scores, *i.e.*, specific recommendation made by the system. Intuitively, this attack involves creating several fake, attacker-controlled "sybil" users whose history is artificially constructed to be very similar to that of the target user. Then the fact that the system is recommending a particular item to the sybils implies with some confidence that this item must have been purchased or rated highly by the target user.

Finally, it is worth mentioning that many recommendation services such as last.fm and librarything.com also make *user similarity lists* available. For each user, these lists explicitly identify other users who have similar histories. By creating sybil users with a particular history or profile, the active attacker can use these similarity lists to infer information about other users who have been identified by the system as similar to the sybils. We defer a detailed analysis of this attack to future work.

Attack metrics. Our attacks are designed to infer information about users' behavior from the output of the recommender system and several types of auxiliary information outlined above. Specifically, they produce *predictions* of the following form: "User X purchased item Y during time period T" (or, in the case of recommendation systems based on explicit ratings rather than purchases, "User X rated item Y"). The two main metrics for measuring the quality of our predictions are yield and accuracy.

Definition 12 (Yield) *The yield of an attack is the number of correct predictions per user per unit time.*

Definition 13 (Accuracy) *The accuracy of an attack is the fraction of predictions that are correct.*

Understanding yield and accuracy. If there are n users, the attacker makes p predictions per time unit, of which q are correct, then the yield is $\frac{q}{n}$ and the accuracy is $\frac{q}{p}$. The accuracy may be very high even if the yield is small: what this means is that the adversary can only occasionally make a prediction, but when he does make one, it is likely to be correct.

There is an inherent tradeoff between the two metrics. When yield increases, the number of false positives (*i.e.*, incorrect predictions) also increases, resulting in a decrease in accuracy. In our experiments, we measure the yield and accuracy for different values of a trade-off parameter, and report the yield-accuracy curve.

Targeted vs. generic attack. There are two subtly different ways in which our prediction algorithm can be used to infer individual transactions. The first is a *targeted attack*. In this attack, the attacker already knows that the target user made a transaction on a certain date. The attacker's objective is to learn which item was purchased or rated. For example, a curious co-worker may notice an empty Amazon box in the hallway and try to find out what product has been bought. This attack results in a higher yield per customer, but to carry it out on a large scale, the attacker needs auxiliary information beyond what is available from the recommender system itself (*e.g.*, it can be feasibly done by a postal employee).

The second attack is the *generic attack*. In this attack, the attacker does not know when transactions occur. His objective is to infer both the fact that a certain customer has performed a transaction and which item was involved in the transaction. The yield of this attack is necessarily lower, but–as we show in our experiments–it can be carried out on a large scale because the auxiliary information needed for this attack is available from the recommender system itself and does not require offline observations of individual customers.

Privacy breach. We emphasize that even though our attack takes advantage of the user's public history of purchases and/or ratings, the predictions made by our algorithm concern the user's *non-public* transactions and thus

constitute a privacy breach. The public history is completely under the user's control: he decides which items to rate and which purchases to reveal. By contrast, the item similarity lists and (in the case of an active attack) user-item recommendation scores revealed by the recommender system are based on the user's *entire* history, including transactions that the user did not disclose voluntarily. The main privacy risk of the public recommender systems, as empirically demonstrated by our algorithms, is the ability to accurately infer these "hidden" transactions.

Disclosure of non-public transactions as a result of analyzing the output of large-scale recommender systems has several privacy implications. First, it may violate the terms of service under which customers share their information with Web retailers and other online providers. For example, the Amazon.com Privacy Notice lists several situations in which it may disclose information about individual customers [4], none of which include revealing a customer's purchases to another customer without consent from the former.

Second, leaking information about individual book purchases via the bookseller's recommender system may violate customers' expectations as well as societal norms that govern the transmission of information between the buyers and online merchants [198].

Finally, unauthorized disclosure of customers' transactions may have legal implications under the Privacy Act of 1974 [251] and, in the case of video material, the Video Privacy Protection Act [87]. Legal analysis is beyond the scope of this work.

Active attacks on recommender systems. There is a literature on *shilling attacks* on collaborative filtering recommender systems [185, 181], where the

aim is not to cause a privacy breach but rather to create sybil users in order to influence the system by "pushing" certain items, causing those items to be recommended more often to users, presumably resulting in financial gain. While our methods are somewhat similar, our goals are very different.

We describe a simple, yet effective attack on the *k-nearest neighbor* (k-NN) recommendation algorithm [9]. Consider the following recommender system. For each user U, it finds k most similar users according to some similarity metric (*e.g.*, the Pearson correlation coefficient or cosine similarity). Next, it ranks all items purchased or rated by one or more of these k users according to the number of times they have been purchased and recommends them to U in this order. We assume that the recommendation algorithm and its parameters are known to the attacker.

Now consider an attacker whose auxiliary information consists of the user U's partial transaction history, *i.e.*, he already knows m items that U has purchased or rated. His goal is to learn U's transactions that he does not yet know about.

The attacker creates k sybil users and populates each sybil's history with the m items that he knows to be present in the target user U's history. Due to the sparsity of a typical transaction dataset, $m \approx O(\log N)$ is sufficient for the attack on an average user, where N is the number of users [194]. (In practice, $m \approx 8$ is sufficient for datasets with hundreds of thousands of users.) With high probability, the k nearest neighbors of each sybil will consist of the other $k - 1$ sybils and the target user U. Therefore, the attacker can simply inspect the list of items recommended by the system to any of the sybils. Any item which appears on the list and is *not* one of the m items from the sybils' artificial history must be an item that U has purchased. Note that

any such item was not previously known to the attacker and learning about it constitutes a breach of U's privacy.

This attack is even more powerful if the attacker is allowed to adaptively change the fake history he constructed for his sybils after observing the output of the recommender system. For example, in the case of movie ratings on Netflix, a user can change previously entered ratings. On Amazon, one can tell the recommender system to ignore certain transactions for the purpose of making recommendations. When this capability is supported, the attack can be feasibly carried out on a large scale with a constant number of sybils.

8.4 Passive attack

Evaluation methodology. While we can and do apply our algorithm directly to Amazon's online recommendation service, measuring the accuracy of our predictions is quite challenging without access to the actual purchase records (which are kept confidential by Amazon, for obvious reasons). As we describe below, it is sometimes possible to verify that the prediction is correct (*e.g.*, the user publicly rates an item after our algorithm has inferred that he or she purchased it), but for the vast majority of predictions, there is no oracle that would tell us whether they were correct.

This presents a methodological problem: how do we prove that our algorithms produce accurate inferences? One of the ways we address this problem is by applying our algorithm to an artificial recommendation service running on top of the Netflix Prize dataset. While the recommendation system is very similar to that used by Amazon, in the case of the Netflix Prize dataset the records of individual Netflix subscribers do tell us which movies they rated.

Therefore, the dataset provides the "ground truth," which can be used to measure the accuracy of our predictions. After our algorithm infers from the output of the recommendation system that a given user has watched a particular movie at a certain time, we simply look at the user's record in the dataset to determine whether the inference is correct or not.

As described in Section 8.3, a passive adversary's power is limited to observing a site and its recommender system. To analyze the power of such an adversary, we implemented an item-to-item recommender system based on the description by Linden *et al.* [165, 166]. We apply the implemented system to the Netflix prize dataset to generate item-to-item recommendations. Based on these recommendations, knowledge of item rankings, and modeled knowledge of past customer purchases, we generate predictions of non-public items purchased by those customers. Two important heuristics help us to narrow the set of predictions to improve accuracy and maintain a strong yield. This section describes our experiments and their results in greater detail.

Netflix dataset versus Amazon dataset. Given the limitations on our access to Amazon's data, the large Netflix prize dataset is a useful alternative. A brief comparison of the datasets may be helpful. Netflix is an online movie rental service, and Netflix customers are subscribers that can rent a limited number of movies at a time. The Netflix prize dataset is a subset of Netflix's customer rating data released for a contest. Although the data was "anonymized" prior to its release, Narayanan and Shmatikov suggest that little change from the true data actually occurred [194]. This Netflix dataset contains approximately 480,000 customers, nearly 18,000 movies, and roughly 100 million rentals by those customers. On an average day, approximately

100,000 rentals occur. While the ratings in the Netflix dataset are not directly associated with purchases, this distinction does not appear important for our analysis (our algorithm ignores the numerical value of ratings).

Amazon is a large online retailer that offers a diverse catalog of products. Based even on 2003 numbers, Amazon offers at minimum millions of items to its tens of millions or more customers [165]. Given that Amazon sold 6.3 million items worldwide on its peak 2008 holiday sales day [14], one may reasonably assume that Amazon sells over 100,000 items per day on average.

By singling out individual features, one could argue that either dataset offers privacy advantages. For example, Amazon's greater number of customers increases the chances that a customer can blend into a crowd with similar purchase patterns, but Netflix's smaller catalog of items can also force customers into crowds.

8.4.1 Prediction algorithm and pseudocode

To understand our prediction algorithm, recall that item-item similarity scores reflect the likelihood of being purchased by the same customers. Thus, if a customer who has previously purchased item I purchases item J, then I and J will increase their similarity scores w.r.t. each other if there is no other change in the system.

Suppose that customer (user) U has made public the previous purchase of item I. On the target date T, he purchases item X. With some probability, X will improve its position in the similarity list of I, or will enter it, not having been on it before. This is by no means a certainty–for random items I and X purchased by U, we do not expect X to be sufficiently similar to I to show up

in the similarity list even after the purchase. Even if X is already in I's list, it might become more similar to I but not by enough to improve its position. Finally, it might not even become more similar, being offset by purchases of either item by customers who have *not* previously purchased the other item.

Nevertheless, U's purchase of X significantly increases the odds that X will "climb" in I's similarity list on target date T. Thus, the attacker's observation of this "climber" is a piece of *supporting evidence* that U purchased X. We call I a *support* for the *prediction* (U, X, T).

Suppose that X climbs simultaneously in the similarity lists of $I_1, I_2, \ldots I_{10}$, all of which have been previously purchased by U. The odds of this happening by chance are quite small; the prediction (U, X, T) is now supported by 10 items. Other than the number of supports, what other factors improve our confidence in a prediction? We have identified two heuristics that are highly effective:

Rarity of supports. Suppose that item I has been purchased only by a single user. Then we know that that user is U. Therefore, only U's other purchases could possibly affect I's similarity list. Thus, we can be *certain* that U purchased X. In general, the less popular a supporting item, the fewer the users whose actions can influence it, and the higher the evidence that X's climb was due to U instead of some other user.

Diversity of supports. Suppose that all the supporting items $I_1, I_2, \ldots I_{10}$ are mutually similar, *i.e.*, they appear on each other's similarity lists. This means that there are potentially many users (with similar interests to the target user) who have all purchased each of the supporting items. On the other hand, if the supporting items are mutually unrelated, it serves as a fingerprint of user U's spectrum of interest. Numerically, the diversity score is computed

as the reciprocal of the number of other supports similar to each support, summed over the supports.

Our notion of diversity is similar to the notion of "straddlers" discussed by Ramakrishnan *et al.* in [216]. Our intuition is that *given enough history, everyone is a straddler*.

That is the essence of our algorithm. It is formally described by the pseudocode below. First it iterates through U's public purchases before date T to compile a list of supporting items for each climber. Then, for each possible prediction X, it computes the rarity and diversity score. If the combined score exceeds a threshold, it outputs (U, X, T).

A higher threshold leads to a higher accuracy, since there is more evidence for each prediction. At the same time, it leads to a lower yield, since fewer predictions are made. By varying the threshold continuously in $[0, \infty)$, we get an accuracy-yield curve.

```
function make_predictions(cust, date, threshold){

  for each possible_support in public_items(cust, date-1){
    for each item in climbers(possible_support, date){
      add possible_support to supports[item]
    }
  }

  // inventory is a list of all items
  for each item in inventory{
    if support_score(supports[item]) >= threshold
      output (cust, item, date)
```

```
  }
}

function support_score(supporting_items){
  rarity_score = sum(1 / (1 + item.popularity) for item in supporting_items)

  function edge?(v1, v2){
    return v1 in v2.similarity_list
  }
  graph = Graph(supporting_items, edge?)
  diversity_score = sum(1 / (1 + graph.degree(item)) for item in supporting_items)

  return rarity_score * diversity_score
}

// returns the items that improved their position in
// the given item's similarity list on the given date
function climbers(item, date){
  ...
}
```

```
// constructs a graph with the given nodes and edge relationships computed from
// the binary predicate edge?
function Graph(nodes, edge?){
  ...
}
```

8.4.2 Experiments

While the Netflix dataset consists of *ratings* rather than purchases, we use the term purchases in order to be consistent in our terminology.

Recommendation algorithm. Our implementation closely follows Amazon's published description [165, 166] and the parameters observed on amazon.com.

Given a target date T, the recommendation algorithm works as follows: for an item I, define $\mathcal{U}_T(I)$ to be the set of customers who have purchased item I at or prior to date T. The similarity score between items I and J is defined as the number of customers who have purchased both items as a fraction of those who have purchased either:

$$\mathsf{Sim}_T(I, J) = \frac{|\mathcal{U}_T(I) \cap \mathcal{U}_T(J)|}{|\mathcal{U}_T(I) \cup \mathcal{U}_T(J)|}$$

This is the Jaccard coefficient, which is closely related to the cosine similarity measure. Our formulation is based on Amazon's patent [166], which says: "A relatively high commonality index for a pair of items item A and item B indicates that a relatively large percentage of users who bought item A also bought item B (and vice versa)." Note that our prediction algorithm

is not very sensitive to the choice of similarity measure, and merely requires that a purchase of an item by a customer makes it more similar to other items previously purchased by that customer.

The item similarity list for each item I consists of the 50 items with the highest value of $\mathsf{Sim}(I, J)$. Our algorithm updates these lists daily, just as they are on amazon.com. The number 50 was chosen because this is the number of related items displayed on Amazon's website for the vast majority of items. Recently, we have observed some items showing up to 100 related items; this would make our attack even more powerful.

We decided not to incorporate numerical ratings in our algorithm, Amazon's recommendations for the most part operate on customer purchases rather than reviews.

We ran this recommendation algorithm for each of the 31 days in the month of July 2005 (the Netflix prize dataset ends at December 2005). Similarity lists for each of the 17,770 items in the dataset were computed, consisting of the 50 items with the highest similarity scores.

Attack. We assume that each customer makes each purchase public independently with a probability of $\frac{1}{3}$ (say, by writing reviews). Our target set consists of the customers who have made at least 100 purchases public by June 30, 2005; there are 77,005 such customers.

A note about the choices of these parameters is in order. As of April 2009, there are around 10,000 customers on amazon.com with 100 or more public reviews. User behavior with regard to reviews ranges from the prolific to the occasional, and we chose $\frac{1}{3}$ as a representative value. The choice of this parameter does not substantially affect the yield per customer.

We run the prediction algorithm on each of the 31 target dates. The

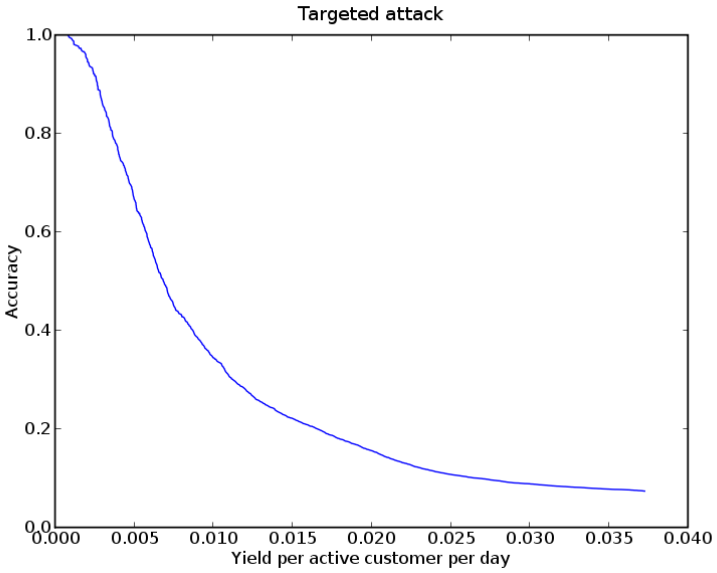

Figure 8.1: Targeted attack

attacker is assumed to have access to

- the public purchases of each customer on each date prior to the target date,
- the similarity lists of each item on each date up to and including the target date, and
- the approximate number of times each item has been purchased up to the target date.[4]

Targeted attack. Recall that in a targeted attack, in addition to the above information, the attacker knows that a specific customer made a purchase on a specific date. Of the 77,005 customers, 57,554 had made at least

[4]Even if these numbers are not directly available, they can be easily approximated from the sales rank data.

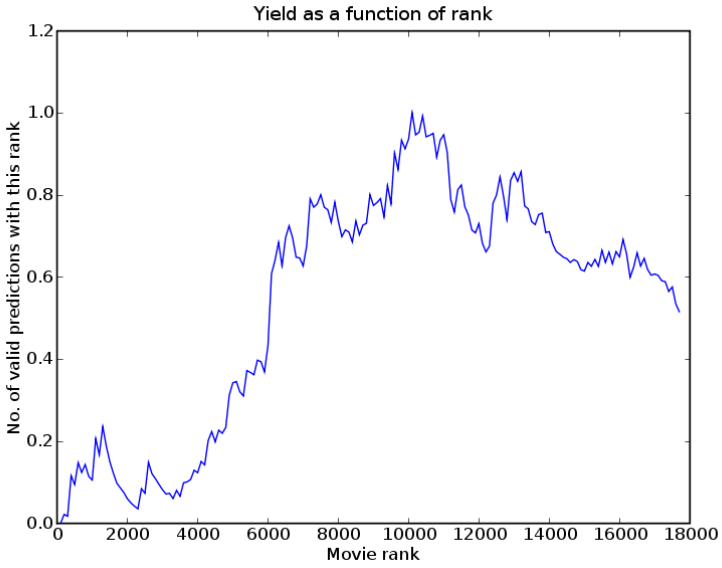

Figure 8.2: Rank distribution

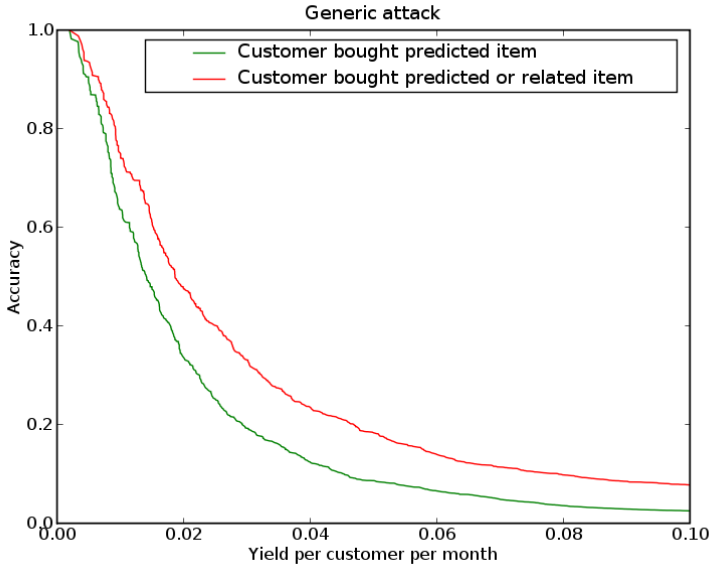

Figure 8.3: Generic attack

one purchase during the month of July 2005 and were therefore potentially vulnerable.

The algorithm made 132,525 predictions during this period, impacting a total of 12,458 customers. The graph of accuracy vs. yield is shown in Figure 8.1. Thus, with a roughly 35% accuracy, a postal service worker can determine the contents of roughly 1 in every 100 boxes shipped from a given online merchant that pass through his hands. This means that a prediction is made only 1% of the time, but conditional on a prediction being made, it is correct 35% of the time.

The predictions made had a median rank of 10,666, which is essentially the same as the median rank of correct predictions (10,533). Recall that the total number of items is 17,770. Less popular items, *i.e.*, items with a higher rank are much more likely to be predicted correctly.[5] Figure 8.2 shows the distribution of yield with item rank. Note that less popular items are generally more sensitive from a privacy perspective.

The graph reports the yield per customer-date pair conditional on a purchase having been made by that customer on that date. The yield per customer per month is not 30 times this number, because not every customer makes a purchase on each day.

Generic attack. As expected, the generic attack has a much lower yield. The yield of this attack per customer scales linearly with time, since it is not limited to the days on which the customer made a purchase; we report the yield for the one-month period of our experiment. The green (lower) curve in Figure 8.3 reports the yield vs. accuracy; for instance, the value of

[5]The peak results from the trade-off between the diminishing number of items with rank X, and the increasing likelihood of an item with rank X being predicted correctly, with increasing X.

(0.4, 0.02) on the line means that with (say) 10,000 vulnerable customers, an attacker can make 500 predictions per month and expect 200 of them to be accurate (the accuracy is $\frac{200}{500} = 0.4$, and the yield is $\frac{200}{10000} = 0.02$).

Many of the incorrect predictions turn out to be not totally off-the-mark, but "close" to the item that the customer purchased on the target date, in the sense that the predicted item is in the similarity list of the target item. This could still be a privacy compromise. The red (upper) curve in Figure 8.3 shows the yield vs. accuracy computed by treating a prediction as correct if it is within the top 20 most similar items of the target item.

8.5 Passive attack on Amazon

Amazon's data. As described in earlier sections, Amazon provides several forms of recommendations. This includes our focus, item-to-item recommendations (also known as similar items or "Customers Who Bought This Item Also Bought ..."). Amazon presents these recommended items every time the customer views an item. We have observed similarity lists including up to 100 items on Amazon's website. These recommendations are computed using similarity scores that appear to be updated daily or perhaps more often, though this may differ based on the item or other factors.

In addition to recommendations, Amazon publishes sales ranks (or item ranks) for items. Sales rank is a measure of the popularity of the item within a given category, such as books, movies, or music. This rank appears to be affected at minimum by the number of customers who have purchased that item and the time elapsed since those purchases. The sales ranks are updated at least hourly.

Amazon allows its customers to review items, and Amazon maintains a publicly accessible a list of tens of thousands of customers designated as "top reviewers" along with links to each customer's reviews. This list is ordered by reviewer rank, which appears to be a product of the number of reviews written, the timing of the reviews, and the number and percentage of other customers who found the reviews helpful. Each reviewer has a unique reviewer identifier in addition to a potentially non-unique rank (*i.e.*, some may be tied). The reviews themselves include the item identifier of the reviewed item, the date of the review, and various means by which the customer can express an opinion of the item. Customers are not required to review items that they purchase, and customers may review items that they did not purchase from Amazon.

Amazon's interfaces. Amazon makes its data available in two ways: via its website and via an API (Amazon Web Services). All item, recommendation, and customer data that Amazon makes available is accessible via the website. The API provides data in a more convenient format but unfortunately provides only a subset of the available data. To abide by various requirements, data collection via the site was performed manually (even though it could be done more efficiently with automated tools) and collection via the API was automated but limited to the request frequencies allowed by Amazon (one request per second).

The API only allows users to collect a customer's oldest 100 reviews. As many customers exceed this quantity of reviews, collection via the website was necessary at certain times. Similarly, although items are associated with up to a hundred similar items via the website, Amazon provides only the top ten similar items via its API. This limitation was particularly problematic for

us as the number of items was too large to allow for meaningful additional manual collection. Therefore, we considered just the top ten similar items.

Data collection. Via Amazon's interfaces, we collected 999 reviewer identifiers. These customers initially occupied a contiguous set of reviewer ranks with none falling in the top 1000. When our recommendation analysis began on March 1, 2008, the corresponding customers had written 120.6 reviews on average (120,452 total). We regularly updated our set of reviews throughout the course of the study. When data collection concluded in July 2008, the average number of reviews had reached 125.6 (125,474 total).

We monitored the recommended items lists daily for all items purchased by a limited subset of our reviewers between March 1, 2008 and March 31, 2008 inclusive. This monitoring was performed via Amazon's API and, therefore, included only the top ten items from these recommendation lists. This subset of monitored reviewers initially contained all customers who had written reviews after August 25, 2007. Over the course of the month, we allowed previously inactive reviewers to join this subset if they reviewed an item. We did not remove any customers from the subset, however. On March 1, we monitored 518 users, and this number had grown to 539 by March 31. We also retrieved limited sales rank data when it could be collected for free with the recommendation list queries.[6]

Making predictions. For our analysis, we modeled an adversary's set of prior knowledge of a customer as all reviews previously written by that customer. Because the set of items purchased may contain items not reviewed and vice

[6]Because any item can move onto and off of recommended item lists, we could not monitor all sales ranks for all possible items for the full month. Fortunately, these ranks can be retrieved without additional queries for all items in a given recommended item list.

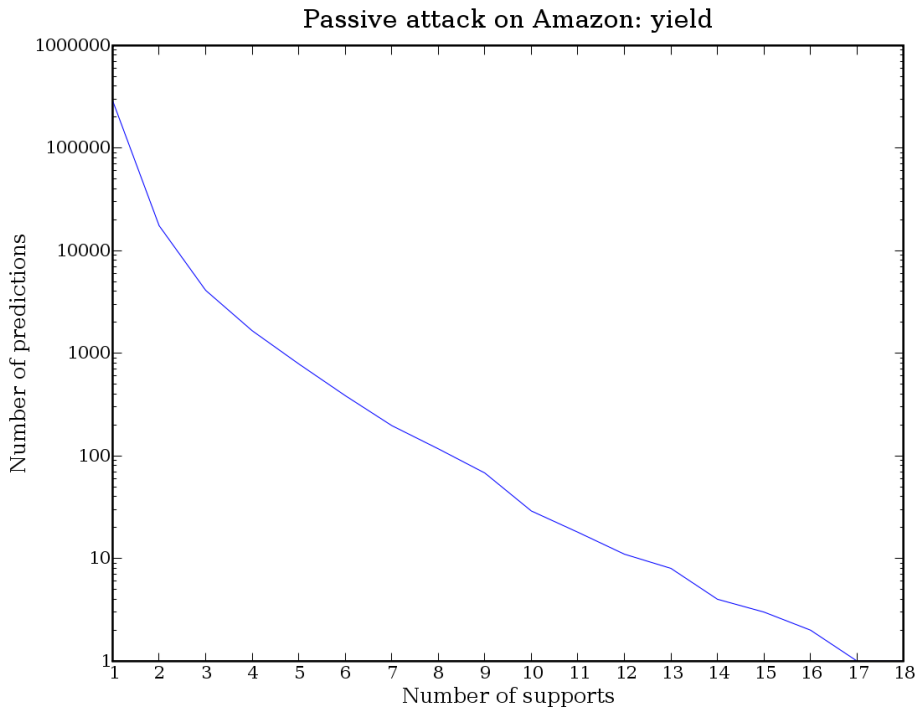

Figure 8.4: Predictions at each level of support

versa, this knowledge is imperfect. We predicted purchases as described above based on which items rose in previously reviewed items' recommendation lists. Because the sales rank data was incomplete, the heuristics used on the Netflix data were not feasible here. An adversary that has a more limited target or is less respectful of Amazon's restrictions could collect a more complete dataset.

The algorithm made a total of 290,182 unique (user, item) predictions based on the month's data. Of these, 787 had at least five supporting items. Figure 8.4 shows the number of predictions generated at or above each level of support. The set of predictions would be larger (and, likely, more accurate) given access to recommended items which are outside the top ten of the list.

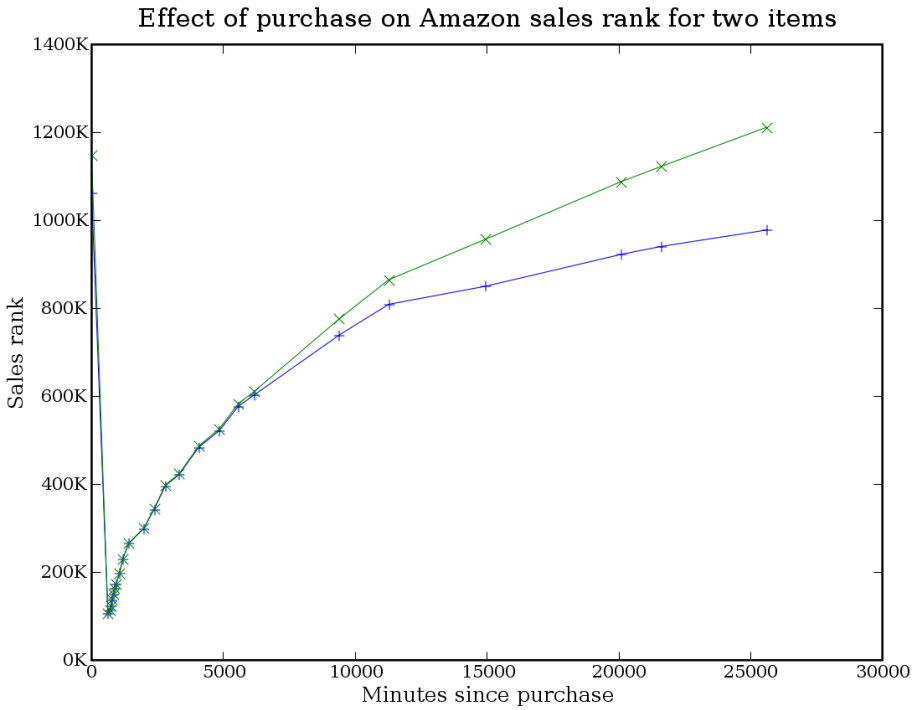

Figure 8.5: Rank change following purchase

Validation methods and discussion. Amazon does not provide purchase data, so we have no means of verifying our predictions; the best we can do is see whether the customer reviewed the predicted item later. We observed our target customers for a two month period after the conculusion of our main data collection phase. During this period, we were able to validate 10 of our predictions which had 5 or more supporting items. Of course, we expect only a fraction of customers to write reviews during our observation period, and so this metric is not a measure of accuracy; it is at best a weak lower bound. Observing a review gives us a high confidence that a prediction is correct, but the lack of a review does not invalidate a prediction.

Similarly, we are attempting to verify our predictions based on what a customer reveals publicly. The more interesting cases from the privacy perspective are the purchases for which the customer would choose not to post a review.

One interesting aspect of Amazon's massive catalog and customer base is that it makes sales rank more useful for narrowing predictions. Suppose that you had previously purchased only item A, and today you purchase item B. This has the same effect on the recommendations as if you had previously purchased only item B and today you purchase item A. More complicated versions of this example exist, and sales rank may help us distinguish between the cases. We would expect sales rank for most items to stay fairly consistent from day to day given such a large number of items and customers. Whichever item was purchased will likely see a slightly boost in sales rank relative to the other, however. The relative boost will be influenced by each item's popularity, *e.g.*, it may be more dramatic if one of the items is extremely unpopular. For example, Figure 8.5 demonstrates how sales rank dramatically improves for

two items after purchase followed by a gradual return toward the original rank. Changes in sales rank appear to be nearly instantaneous, potentially making even simple timing attacks feasible. Given access to a complete set of sales rank data, an analysis of its security implications could be beneficial.

Case studies. We present three case studies. These are based on our data, but some details have been removed. To avoid confusion, the predicted item is labeled A in all cases, and the supporting items are labeled B, C, and so on.

William Smith (not real name) is a regular reviewer, having written over 100 product reviews by March 1, 2008. Many of these reviews were of gay-themed books and movies. Product A is a gay-themed movie available on Amazon. On March 20, the sales rank for this product was just under 50,000, but it had jumped to under 20,000 by March 21. Mr. Smith's previous reviews included products B, C, D, E, and F. Product A was not recommended in the top ten for any of these five items on March 19 but had moved onto the top ten list for all five of them by March 20. Based on the recommendation information, our algorithm predicted that Mr. Smith had purchased product A. Within a month of this jump, Mr. Smith reviewed product A.

Jane Brown (not real name) is also a regular reviewer that has commented on several R&B albums in the past. Item A is an older R&B album. On March 1, the album had a sales rank of over 70,000, but that rank had decreased to under 15,000 by March 2. Ms. Brown had previously reviewed items B, C, and D, among others. Item A moved into item B and item C's recommendation lists on March 2, and it rose higher in item D's list that same day. Based on this recommendation information, our algorithm predicted that

Ms. Brown had purchased product A. Within two months of this activity, Ms. Brown reviewed item A.

Mark Grant (not real name) appears to be interested in action and fantasy stories. He is a regular Amazon reviewer as well. Product A is fairly recent a fantasy-themed movie. On March 18, the product jumped from in sales rank from slightly under 35,000 to under 15,000. It experienced another minor jump the following day, indicating another potential purchase. Mr. Grant's previous reviews included items B, C, and D. On March 19, product A rose in the first two items' similarity lists and entered the top ten of the final item's list. None of the supporting items have sales rank changes that indicate purchases on that date. Based on the recommendation information, our algorithm predicted that Mr. Grant had bought product A. Within one month of this activity, Mr. Grant reviewed product A.

In all cases, the reviewers are clearly comfortable with public disclosure of their purchases, since they ultimately reviewed the items. Nevertheless, our success in these cases suggests a realistic possibility that sensitive purchases can be inferred. While these examples include predictions supported by items in a similar genre, we have observed cross-domain recommendations on Amazon, and much anecdotal evidence suggests that revealing cross-domain predictions are possible. For example, Fortnow [100] points to a case in which an Amazon customer's past opera purchases resulted in a recommendation for a book of suggestive male photography.

8.6 Defenses

Defenses against inference attacks on recommender systems techniques should balance the goals of allowing timely, high-quality recommendations and protecting against meaningful inference of individual user purchases.

The most effective mitigation strategy is to integrate privacy into the collaborative filtering process in a way that offers provable privacy guarantees. McSherry and Mironov have recently developed a technique for doing so based on a relaxed definition of differential privacy [178]. Their approach is to first compute an approximation to the item-item covariance matrix (and a few other "global" statistics) in a differentially private way. This guarantees that any further computations that depend only on this data also result in differentially private outputs. There are a number of geometric algorithms, such as Singular Value Decomposition, that can be evaluated using the differentially privately computed statistics.

Although this approach limits the set of collaborative filtering algorithms that may be used, and the accuracy of the output is somewhat diminished, the authors evaluate their approach on the Netflix Prize dataset and find that an accuracy that is superior to the Cinematch baseline can be achieved while providing reasonable privacy guarantees.

McSherry and Mironov do not address the issue of time-dependent recommendations, *i.e.*, preserving privacy guarantees while updating frequently recommendations, say every day, as Amazon does. While differential privacy does guarantee "composability," the numerical privacy bounds degrade rapidly (linearly) with time. Nevertheless, this approach currently offers by far the best hope for a sound defense against our attack.

We now discuss several other heuristic mitigation techniques.

- *Limit the length of recommended items lists.* Amazon's "Customers Who Bought This Item Also Bought ..." list can contain one hundred items and perhaps more. Recommendations near the top of an item's list have a strong relationship with that item. A single purchase is unlikely to impact this relationship significantly. Recommendations near the bottom of the list have weaker relationships with the item and, therefore, have a more volatile ordering.

- *Factor item popularity into update frequency.* Less popular items tend to be more uniquely identifying, so limiting the frequency of updates involving these items might decrease the amount of information that can leak. For example, Amazon might add purchases of unpopular items to the item-item matrix only once per week or even less often. In addition, sales rank for these items may betray when they are purchased, so sales rank may benefit from greater inertia. Unfortunately, this may dull the impact of sudden, important shifts in the data.

- *Allow users to opt out.* Amazon allows customers to request that certain purchases not be used in personalized recommendations to themselves. (This option primarily prevents gift purchases for others from influencing recommendations.) For a customer concerned with privacy, a more useful option would be to prevent a purchase from influencing the recommender system in any manner whatsoever. If users habitually chose to opt out, recommendation quality could suffer significantly. Therefore, the feasibility of this option depends on the system design and user behavior.

- *Avoid cross-genre recommendations.* If customers with interests in multiple genres tend to be at higher risk [216], this risk could be mitigated by avoiding cross-genre recommendations except when items have strong relationships. This has the shortcoming of obstructing potentially surprising but useful recommendations.

- *Limit the speed of data access.* Large-scale passive attacks require that an adversary monitor a reasonably large quantity of data. Limiting speed of access (by rate-limiting the API, using crawler-detection heuristics, *etc*) may reduce the amount of data that an adversary can monitor and, consequently, the scale of the attack. Unfortunately, this may prevent legitimate uses of the data in some cases. In addition, these limits may not stop smaller, more-focused attacks or a determined, capable attacker (*e.g.*, one with access to a botnet).

While each mitigation strategy has limitations, a careful combination of them may provide substantial practical benefits with only modest drawbacks.

8.7 Summary

Recommender systems have become an essential part of the business strategies of many online retailers. The attacks developed in this work demonstrate that these systems can also leak information about individual user actions, posing a serious risk to user privacy. Our primary attack can be performed by a purely passive adversary without any special access to the system, making this attack especially powerful and dangerous. The attack utilizes item-item recommendations, which are available from Amazon and many other online

retailers. A number of defenses are possible, but no known solution guarantees privacy without degrading the quality of recommendations.

Our focus is on larger, established e-commerce sites, but our attack is also applicable to smaller or newer sites, potentially posing an even greater risk. While risk depends on a number of factors, the datasets of smaller, less-established retailers tend to be less stable, increasing the likelihood that a single user action yields a noticeable impact.

This work undermines the perceived dichotomy between individual user data and large-scale, aggregate recommendation data. Recommender systems produce an output which has a complex, not fully understood dependence on the inputs. Our research underscores the need for a coherent framework for analyzing the effects that an individual user's actions have as they merge with other data to create recommendations.

Chapter 9

Conclusions

The picture that is emerging from the above work is a bleak one for non-interactive data sharing. Focusing on the problem of anonymous sharing of high-dimensional data, we do not believe that there exists a technical solution to be found. Specifically, we do not believe that there are anonymization strategies that can (a) satisfy a robust definition of privacy, (b) withstand de-anonymization attacks described in our paper, and (c) preserve utility for common data-mining purposes. Therefore, we advocate non-technical solutions.

First, the false dichotomy between personally identifiable and non-personally identifiable information should disappear from privacy policies, laws, etc. Any aspect of an individual's online personality can be used for de-anonymization, and this reality should be recognized by the relevant legislation and corporate privacy policies. Laws, regulations, and business practices need to be adjusted to reflect the fine gradations of intents and purposes underlying information sharing and disclosure.

Second, data collectors should stop relying on anonymization as the "get

out of jail" card insofar as user privacy is concerned. They should inform users when their information is disclosed to third parties, even if this information has been anonymized, and give them an opportunity to opt out. We expect that most users will not object to their information being used for commercial purposes such as targeted advertising. Furthermore, service providers who only deal with reputable advertisers and marketing partners may even enjoy a competitive advantage in the marketplace.

In the realm of technical solutions, it appears that the future must involve practitioners transitioning to the interactive setting. We propose a multi-layered approach to make this possible: first more research. It is known that the SuLQ model, singular value decomposition, k-means clustering, the perceptron algorithm and the ID3 classifier can all be computed privately in that setting [35]. The more complex practical applications that can be privately computed, increases the chance privacy preserving data mining applications with provable guarantees will become practically feasible. Second, more programming tools like PINQ make it possible translate these theoretical results into working systems [177], [176]. Finally, more investment in infrastructure is needed, such as the Amazon Elastic Compute cloud that make it easier data collectors who don't possess much infrastructure on their own to provide such collaborative computing services.

Appendix A

Regular expressions for common password patterns

To simplify the regular expressions as well as the indexing algorithms, we specify length restrictions separately. Each regular expression has the input alphabet $\{A, a, n, s\}$, representing uppercase, lowercase, numeral and special characters, respectively. For each regular expression, we specify 4 numbers, which represent how many times the corresponding symbol type is permitted to occur in any accepted string. We also specify whether Markovian filtering is to be used, and if so, whether it should be zero- or first-order. Finally, we specify the threshold (recall that with Markovian filtering, we will only consider strings whose Markovian weight is above the threshold). The threshold is specified by stating what fraction of the keyspace should belong to the dictionary. The last column in the table is the size of the keyspace. It is not part of the keyspace specification. The size of the combined keyspace is the sum of the entries in the last column.

Determining the threshold for each regular expression is somewhat subjective, but we followed these general rules: no keyspace can have a size more than 10^8; the dictionary size should be 10% of the keyspace (based on Markovian filtering) unless the number of characters is 4 or less; in the latter case it should be 30% of the keyspace.

Regexes with 5 characters or less are not shown because they contribute too little to the keyspace size. We use the first-order Markov model whenever there are at least 6 alphabetical characters, all of them contiguous.

Table A.1: Regular expressions

Regex	A	a	n	s	Fraction	Markov	Size
a*	0	8	0	0	0.000478	1	10^8
a*	0	7	0	0	0.0124	1	10^8
a*	0	6	0	0	0.1	1	3.09×10^7
A*	8	0	0	0	0.000478	1	10^8
A*	7	0	0	0	0.0124	1	10^8
A*	6	0	0	0	0.1	1	3.09×10^7
[Aa]*	1	6	0	0	0.00178	1	10^8
[Aa]*	1	5	0	0	0.0540	1	10^8
n*	0	0	8	0	1	-	10^8
n*	0	0	7	0	1	-	10^7
n*	0	0	6	0	1	-	10^6
a*n*	0	6	1	0	0.0324	1	10^8
a*n*	0	5	1	0	0.1	0	1.18×10^7
a*n*	0	4	2	0	0.3	0	1.37×10^7
a*n*	0	3	3	0	1	-	1.75×10^7
a*n*	0	2	4	0	1	-	6.76×10^6
a*n*	0	1	5	0	1	-	2.6×10^6
a*n*	0	1	6	0	1	-	2.6×10^7
A*n*	6	0	1	0	0.0324	1	10^8
A*n*	5	0	1	0	0.1	0	1.18×10^7
A*n*	4	0	2	0	0.3	0	1.37×10^7
A*n*	3	0	3	0	1	-	1.75×10^7
A*n*	2	0	4	0	1	-	6.76×10^6
A*n*	1	0	5	0	1	-	2.6×10^6
A*n*	1	0	6	0	1	-	2.6×10^7
a*A	1	6	0	0	0.0124	1	10^8
a*A	1	5	0	0	0.1	1	3.09×10^7

Table A.2: Regular expressions

Regex	A	a	n	s	Fraction	Markov	Size
Aa*	1	6	0	0	0.0124	1	10^8
Aa*	1	5	0	0	0.1	1	3.09×10^7
Aa*n	1	5	1	0	0.0324	1	10^8
Aa*n	1	4	1	0	0.1	0	1.18×10^7
[An]*	5	0	1	0	0.1	0	7.13×10^7
[An]*	4	0	2	0	0.1	0	6.85×10^7
[An]*	3	0	3	0	0.284	0	10^8
[An]*	2	0	4	0	0.3	0	3.03×10^7
[An]*	1	0	5	0	1	0	1.56×10^7
[an]*	0	5	1	0	0.1	0	7.13×10^7
[an]*	0	4	2	0	0.1	0	6.85×10^7
[an]*	0	3	3	0	0.284	0	10^8
[an]*	0	2	4	0	0.3	0	3.03×10^7
[an]*	0	1	5	0	1	0	1.56×10^7
[As]*	6	0	0	1	0.0108	0	10^8
[As]*	5	0	0	1	0.1	0	2.97×10^7
[As]*	4	0	0	2	0.1	0	1.7×10^7
[As]*	3	0	0	3	1	-	4.38×10^7
[As]*	2	0	0	4	1	-	6.34×10^6
[As]*	1	0	0	5	1	-	4.88×10^5
[as]*	0	6	0	1	0.0108	0	10^8
[as]*	0	5	0	1	0.1	0	2.97×10^7
[as]*	0	4	0	2	0.1	0	1.7×10^7
[as]*	0	3	0	3	1	-	4.38×10^7
[as]*	0	2	0	4	1	-	6.34×10^6
[as]*	0	1	0	5	1	-	4.88×10^5

Appendix B

Glossary: De-anonymization of sparse datasets

Table B.1: Glossary: De-anonymization of sparse datasets

Symbol	Meaning
D	Database
$\hat{D}$	Released sample
N	Number of rows
M	Number of columns
m	Size of aux
X	Domain of attributes
$\perp$	Null attribute
$\mathsf{supp}(.)$	Set of non-null attributes in a row/column
Sim	Similarity measure
Aux	Auxiliary information sampler
aux	Auxiliary information
Score	Scoring function
ϵ	Sparsity threshold
δ	Sparsity probability
θ	Closeness of de-anonymized record
ω	Probability that de-anonymization succeeds
r, r'	Record
Π	P.d.f over records
H_S	Shannon entropy
H	De-anonymization entropy
ϕ	Eccentricity

Appendix C

Glossary: Social networks

Basic terms.

- S: a social network, consisting of:
  - G: a graph containing nodes V and edges E
  - $\mathcal{X}$: a set of node attributes
  - $\mathcal{Y}$: a set of edge attributes
- X: a node attribute, part of $\mathcal{X}$.
- $X[v]$: the value of the attribute X on the node v
- Y: an edge attribute, part of $\mathcal{Y}$.
- $Y[e]$: the value of the attribute Y on the edge e
- PP: a privacy policy

Sanitized and auxiliary data

- S_{san}: a sanitized social network, defined analogously.

- G_{san}, a sanitized graph, containing $V_{\mathsf{san}} \subset V$ and E_{san}, a noisy version of E
- S_{aux}: the attacker's aggregate auxiliary information, consisting of
  - $G_{\mathsf{aux}} = (V_{\mathsf{aux}}, E_{\mathsf{aux}})$
  - $\mathsf{Aux} = \mathsf{Aux}_X \cup \mathsf{Aux}_Y$, (probabilistic) auxiliary information about node and edge attributes
- $\mathsf{Aux}[X, v]$: the probability distribution of the attacker's knowledge of the value of the attribute X on the node v
- $\mathsf{Aux}[Y, e]$: likewise for edge attributes

Re-identification

- $\mu_G(.)$: ground truth, a 1-1 mapping between V_{aux} and V_{san}
- $\tilde{\mu}(.,.)$: a probabilistic mapping output by a re-identification algorithm
- $\mu(.)$: a specific mapping between V_{aux} and V_{san} sampled from $\tilde{\mu}$
- $\nu(v)$: node centrality (Section 7.4.6).
- α_V: node overlap between V_{aux} and V_{san} (Section 7.6.2)
- α_E: edge overlap between E_{aux} and E_{san} projected on V_{mapped} (Section 7.6.2)
- ϵ: noise parameter (for seed identification)
- β: noise parameter (for propagation; Section 7.6.2)

Appendix D

Measuring the effect of perturbation

The Jaccard Coefficient can be used to measure the amount of perturbation introduced to the sanitized graph S_{san} during the release process:

$$\frac{\sum_{u \in V_{\mathsf{san}}} \nu(u) JC(u)}{\sum_{u \in V_{\mathsf{san}}} \nu(u)}$$

where $\nu(u)$ is the centrality of the node u and the Jaccard Coefficient $JC(u)$ is defined in this context as follows:

$$\frac{|\{v \in \tilde{V} \colon (E(u,v) \wedge \tilde{E}(u,v)) \vee (E(v,u) \wedge \tilde{E}(v,u))\}|}{|\{v \in \tilde{V} \colon E(u,v) \vee \tilde{E}(u,v) \vee E(v,u) \vee \tilde{E}(v,u)\}|}$$

where $\tilde{V} = V_{\mathsf{san}}$ and $\tilde{E} = E_{\mathsf{san}}$. In the above expression, the numerator counts the number of edges that are left unchanged in E_{san}, taking directionality into account. The denominator counts all edges that exist in either direction in either E, or E_{san}.

A more obvious measure that simply counts the number of edges added or removed, as a fraction of the total number of edges, would ignore the effect of perturbation on individual nodes. By contrast, our measure takes this into account, weighing nodes in proportion to their centrality in the network (this is the purpose of the ν factor).

Appendix E

Glossary: Collaborative filtering

User-to-item recommendations. The traditional notion of a recommender system, where products are suggested to customers.

User similarity list. A list of users (customers) whose behavior is most similar to a given user.

User-based recommendations. User-to-item recommendations computed using user similarity lists (or similarity scores).

User-to-user recommendations. The practice of suggesting users (customers) to other customers, *i.e.*, revealing the user similarity list.

Item similarity list. A list of items (products) that have the most similar set of customers who purchased them.

Item-based recommendations. User-to-item recommendations computed using item similarity lists, by aggregating over the item similarity lists of all the items that a customer has previously purchased.

Item-to-item recommendations. The practice of suggesting items that are similar to an item being viewed or purchased, *i.e.*, revealing the item similarity list.

Note. Linden *et al.* [165] use "item-to-item collaborative filtering" synonymously with "item-based recommender system."

Bibliography

[1] MD5 online cracking using rainbow tables. http://www.passcracking.com/.

[2] Microformats. http://microformats.org/.

[3] OpenID. http://openid.net/.

[4] Amazon.com privacy notice. http://www.amazon.com/gp/help/customer/display.html?ie=UTF8&nodeId=468496, 2008.

[5] J. Abello, P. Pardalos, and M. G. C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, *External memory algorithms and visualization*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, 2001.

[6] N. Adam and J. Worthmann. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[7] Add Health. Deductive disclosure. http://www.cpc.unc.edu/projects/addhealth/data/dedisclosure.

[8] The National Longitudinal Study of Adolescent Health. http://www.cpc.unc.edu/projects/addhealth.

[9] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[10] C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proc. 31st International Conference on Very Large Data Bases (VLDB)*, pages 901–909. VLDB Endowment, 2005.

[11] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 247–255. ACM, 2001.

[12] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 86–97. ACM, 2003.

[13] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 439–450. ACM, 2000.

[14] Amazon.com, Inc. Amazon.com's 14th holiday season is best ever. http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=1239175 (Accessed Feb 2, 2009).

[15] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 7–15. ACM, 2008.

[16] C. Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.

[17] C. Anderson. Social networking is a feature, not a destination. `http://www.thelongtail.com/the_long_tail/2007/09/social-networki.html`, 2007.

[18] M. Anderson. Mining social connections. Adweek. `http://tinyurl.com/6768nh`, 2008.

[19] S. Aral. Sexual network patterns as determinants of STD rates: Paradigm shift in the behavioral epidemiology of STDs made visible. *Sexually Transmitted Diseases*, 26(5):262–264, 1999.

[20] M. Arrington. Don't post the evidence unless it supports your case. Techcrunch. `http://tinyurl.com/6otok7`, 2008.

[21] D. Aucsmith. Tamper resistant software: an implementation. In *Proc. First International Workshop on Information Hiding*, volume 1174 of *LNCS*, pages 317–333. Springer, 1996.

[22] Amazon Web Services. `http://aws.amazon.com/`.

[23] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou R3579X?: Anonymized social networks, hidden patterns, and structural steganog-

raphy. In *Proc. 16th International Conference on World Wide Web (WWW)*, pages 181–190. ACM, 2007.

[24] N. T. Bailey. *The Mathematical Theory of Infectious Diseases (2nd edition)*. Hafner Press, 1975.

[25] W. E. Baker. Market networks and corporate behavior. *American Journal of Sociology*, 96:589–625, 1990.

[26] A-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[27] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Proc. Advances in Cryptology - CRYPTO*, volume 2139 of *LNCS*, pages 1–18. Springer, 2001.

[28] P. S. Bearman, J. Moody, and K. Stovel. Chains of affection: The structure of adolescent romantic and sexual networks. *American Journal of Sociology*, 110(1):44–91, 2004.

[29] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *J. Cryptology*, 10(1):17–36, 1997.

[30] R. M. Bell, Y. Koren, and C. Volinsky. The BellKor solution to the Netflix Prize. `http://www.research.att.com/~volinsky/netflix/ProgressPrize2007BellKorSolution.pdf`, 2009.

[31] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proc. Advances in Cryptology – EUROCRYPT*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.

[32] S. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *Proc. IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society, 1992.

[33] S. Bellovin and M. Merritt. Augmented encrypted key exchange. In *Proc. First ACM Conference on Computer and Communications Security (CCS)*, pages 244–250. ACM, 1993.

[34] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proc. International Conference on Database Theory*, pages 217–235. Springer-Verlag, 1999.

[35] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *Proc. 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 128–138. ACM, 2005.

[36] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proc. 40th ACM Symposium on Theory of Computing (STOC)*, pages 609–618. ACM, 2008.

[37] P. Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[38] P. Bonacich and P. Lloyd. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 23(3):191–201, 2001.

[39] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Proc. Advances in Cryptology - EUROCRYPT*, volume 3027 of *LNCS*, pages 506–522. Springer, 2004.

[40] A. Booker. The Nth prime algorithm. `http://primes.utm.edu/nthprime/algorithm.php`, 2005.

[41] S.A. Boorman. A combinatorial optimization model for transmission of job information through contact networks. *Bell Journal of Economics*, 6(1):216–249, 1975.

[42] A. Borges. Toward a new supermarket layout: From industrial categories to one stop shopping organization through a data mining approach. In *Proc. SMA Retail Symposium*, 2003.

[43] J. Borst, B. Preneel, and J. Vandewalle. On the time-memory trade-off between exhaustive key search and table precomputation. In *Proc. 19th Symposium on Information Theory in the Benelux*, pages 111–118. Werkgemeenschap Informatie en Communicatietheorie, 1998.

[44] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Proc. Advances in Cryptology – EUROCRYPT*, volume 1807 of *LNCS*, pages 156–171. Springer, 2000.

[45] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 70–78. ACM, 2008.

[46] E. Brynjolfsson, Y. Hu, and M. Smith. Consumer surplus in the digital economy. *Management Science*, 49(11), 2003.

[47] W. Burr, D. Dodson, and W. Polk. Electronic authentication guideline. NIST Special Publication 800-63, 2004.

[48] J. Calandrino, A. Narayanan, E. Felten, and V. Shmatikov. Don't review that book: Privacy risks of collaborative filtering. Manuscript, 2009.

[49] California Senate Bill 1386. `http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html`, 2002.

[50] California Codes. Business and Professions Code Section 22575-22579. `http://tinyurl.com/5fu9ks`, 2003. Commonly known as the Online Privacy Protection Act of 2003.

[51] A. Campan and T. Truta. A clustering approach for data and structural anonymity in social networks. In *Proc. 2nd SIGKDD International Workshop on Privacy, Security, and Trust in KDD (PinKDD)*. ACM, 2008.

[52] R. Canetti. Towards realizing random oracles: hash functions that hide all partial information. In *Proc. Advances in Cryptology - CRYPTO*, volume 1294 of *LNCS*, pages 455–469. Springer, 1997.

[53] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. In *Proc. 30th ACM Symposium on Theory of Computing (STOC)*, pages 131–140. ACM, 1998.

[54] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proc. 25th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 238–245. ACM Press, 2002.

[55] R. Carthy. Will IRSeeK have a chilling effect on IRC chat? `http://www.techcrunch.com/2007/11/30/will-irseek-have-a-chilling-effect-on-irc-chat/`, 2007. [Note: A privacy outcry erupted over a search engine for (public) IRC channels].

[56] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Towards privacy in public databases. In *Proc. 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *LNCS*, pages 363–385. Springer, 2005.

[57] M. Chew, D. Balfanz, and B. Laurie. (Under)mining privacy in social networks. In *Proc. Web 2.0 Security & Privacy (W2SP)*. IEEE, 2008.

[58] S. Chiasson, R. Biddle, and P. C. van Oorschot. A second look at the usability of click-based graphical passwords. In *Proc. 3rd Symposium on Usable Privacy and Security (SOUPS)*, pages 1–12. ACM, 2007.

[59] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[60] S. Chow, P. Eisen, H. Johnson, and P. van Oorschot. White-box cryptography and an AES implementation. In *9th International Workshop on Selected Areas in Cryptography (SAC)*, volume 2595 of *LNCS*, pages 250–270. Springer, 2003.

[61] S. Chow, P. Eisen, H. Johnson, and P. van Oorschot. A white-box DES implementation for DRM applications. In *ACM Digital Rights Management Workshop*, volume 2696 of *LNCS*, pages 1–15. Springer, 2003.

[62] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. k-anonymity. In T. Yu and S. Jajodia, editors, *Secure Data Management in Decentralized Systems*, volume 33 of *Advances in Information Security*, pages 323–353. Springer, 2007.

[63] S. Clifford. Web privacy on the radar in Congress. New York Times. `http://tinyurl.com/6l2tcw`, 2008.

[64] C. Collberg and C. Thomborson. Watermarking, tamper-proofing, and obfuscation - tools for software protection. *IEEE Transactions on Software Engineering*, 28(8):735–746, 2002.

[65] C. Collberg, C. Thomborson, and D. Low. A taxonomy of obfuscating transformations. Technical Report 148, Department of Computer Sciences, The University of Auckland, 1997.

[66] C. Collberg, C. Thomborson, and D. Low. Manufacturing cheap, resilient, and stealthy opaque constructs. In *Proc. 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 184–196. ACM, 1998.

[67] The Connecticut District Telephone Company. The telephone directory. Reproduced at `http://www.christies.com/LotFinder/lot_details.aspx?intObjectID=5084352`, 1878.

[68] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 160–168. ACM, 2008.

[69] T. Dalenius. Finding a needle in a haystack—or identifying anonymous census record. *Journal of Official Statistics*, 3:329–336, 1986.

[70] H. D'Andrade. MySpace and Facebook plan to use personal data for "targeted advertising". `http://tinyurl.com/2yp7br`, 2007.

[71] The DataPortability project. `http://dataportability.org`.

[72] D. Davis, F. Monrose, and M. Reiter. On user choice in graphic password schemes. In *Proc. 13th USENIX Security Symposium*, pages 151–164. USENIX, 2004.

[73] D. Dean and A. Stubblefield. Using client puzzles to protect TLS. In *Proc. 10th USENIX Security Symposium*, pages 1–8. USENIX, 2001.

[74] D. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.

[75] M. Deshpande and G. Karypis. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[76] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Proc. 2nd International Workshop on Privacy Enhancing Technologies (PET)*, volume 2482 of *LNCS*, pages 54–68. Springer, 2003.

[77] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proc. 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 202–210. ACM, 2003.

[78] Y. Dodis and A. Smith. Correcting errors without leaking partial information. In *Proc. 37th ACM Symposium on Theory of Computing (STOC)*, pages 654–663. ACM, 2005.

[79] J. Domingo-Ferrer and J. Mateo-Sanz. Current directions in statistical data protection. *Research in Official Statistics*, 14(2):105–112, 1998.

[80] R.I.M. Dunbar. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 22(3):469–493, 1992.

[81] C. Dwork. Differential privacy. In *Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP), Part II*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006.

[82] C. Dwork. Differential privacy: A survey of results. *Theory and Applications of Models of Computation—TAMC*, 4978:1–19, 2008.

[83] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proc. 3rd Theory of Cryptographic Conference (TCC)*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

[84] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proc. Advances in Cryptology - CRYPTO*, volume 740 of *LNCS*, pages 139–147. Springer, 1993.

[85] E. Eldon. VentureBeat: Adisn, another company that uses social data to target ads, raises $1.6 million. http://tinyurl.com/65lsd5, 2008.

[86] E. Eldon. VentureBeat: MediaSixDegrees targets ads using social graph information. http://tinyurl.com/662q3o, 2008.

[87] Electronic Privacy Information Center. The Video Privacy Protection Act (VPPA). http://epic.org/privacy/vppa/.

[88] European Parliament. Directive 95/46/EC. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML, 1995.

[89] A. Evfimevski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 211–222. ACM, 2003.

[90] Facebook. Facebook's privacy policy. http://www.new.facebook.com/policy.php, 2007.

[91] Facebook. Statistics. http://www.facebook.com/press/info.php?statistics, 2009.

[92] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. Secure multiparty computation of approximations. In *Proc. 28th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of *LNCS*, pages 927–938. Springer, 2001.

[93] D. C. Feldmeier and P. R. Karn. UNIX password security - Ten years later. In *Proc. Advances in Cryptology - CRYPTO*, volume 435 of *LNCS*, pages 44–63. Springer, 1989.

[94] A. Felt and D. Evans. Privacy protection for social networking APIs. In *Proc. Web 2.0 Security & Privacy (W2SP)*, 2008.

[95] A. Fiat and M. Naor. Rigorous time/space tradeoffs for inverting functions. In *Proc. 23rd ACM Symposium on Theory of Computing (STOC)*, pages 534–541. ACM, 1991.

[96] S. E. Fienberg, U. E. Makov, and R. J. Steele. Disclosure limitation using perturbation and related methods for categorical data (with discussion). *Journal of Official Statistics*, 14:485–502, 1998.

[97] B. Fitzpatrick and D. Recordon. Thoughts on the social graph. `http://bradfitz.com/social-graph-problem/`, 2007.

[98] D. Fono and K. Raynes-Goldie. Hyperfriends and beyond: Friendship and social norms on LiveJournal. In *Internet Research Volume 4: Selected Papers from the Association of Internet Researchers Conference*, 2007.

[99] G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[100] L. Fortnow. Computational complexity: Outed by Amazon. `http://weblog.fortnow.com/2008/02/outed-by-amazon.html` (Accessed Feb 4, 2009), 2008.

[101] M. Franklin and D. Malkhi. Auditable metering with lightweight security. *J. Computer Security*, 6(4):237–255, 1998.

[102] D. Frankowski, D. Cosley, S. Sen, L. Terveen, and J. Riedl. You are what you say: privacy risks of public mentions. In *Proc. 29th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 565–572. ACM, 2006.

[103] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *Proc. 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *LNCS*, pages 303–324. Springer, 2005.

[104] L. Freeman. *The Development of Social Network Analysis*. Empirical Press, 2004.

[105] K. Frikken and P. Golle. Private social network analysis: How to assemble pieces of a graph privately. In *Proc. 5th ACM Workshop on Privacy in Electronic Society (WPES)*. ACM, 2006.

[106] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 265–273. ACM, 2008.

[107] R. Garfinkel, R. Gopal, B. Pathak, R. Venkatesan, and F. Yin. Empirical analysis of the business value of recommender systems, 2006. `http://ssrn.com/abstract=958770`.

[108] R. Gennaro. Faster and shorter password-authenticated key exchange. In *Proc. 5th Theory of Cryptography Conference (TCC)*, volume 4948 of *LNCS*, pages 589–606. Springer, 2008.

[109] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. *ACM Transactions on Information Systems Securrity*, 9(2):181–234, 2006.

[110] C. Gentry, P. MacKenzie, and Z. Ramzan. PAK-Z+. *DoCoMo USA Labs Technical Report*, 2005.

[111] C. Gentry, P. MacKenzie, and Z. Ramzan. Password authenticated key exchange using hidden smooth subgroups. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS)*, pages 299–309. ACM, 2005.

[112] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. 30th ACM Symposium on Theory of Computing (STOC)*, pages 151–160. ACM, 1998.

[113] O. Goldreich and Y. Lindell. Session-key generation using human random passwords. In *Proc. Advances in Cryptology - CRYPTO*, volume 2139 of *LNCS*, pages 408–432. Springer, 2001.

[114] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, pages 218–229. ACM, 1987.

[115] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[116] P. Golle and D.Wagner. Cryptanalysis of a cognitive authentication scheme. In *Proc. IEEE Symposium on Security and Privacy*, pages 66–70. IEEE Computer Society, 2007.

[117] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Pervasive '09 (to appear)*, 2009.

[118] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. 13th ACM Conference on Computer and Communications Security (CCS)*, pages 89–98. ACM, 2006.

[119] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(1360-80), 1973.

[120] Mark Granovetter. Economic action and social structure: The problem of embeddedness. *American Journal of Sociology*, 91:481–510, 1985.

[121] T. L. Griffiths and J. B. Tenenbaum. Probability, algorithmic complexity, and subjective randomness. In *Proc. 25th Conference of the Cognitive Science Society*, 2003.

[122] T. L. Griffiths and J. B. Tenenbaum. From algorithmic to subjective randomness. In *Advances in Neural Information Processing Systems*. MIT Press, 2004.

[123] R. Gross, A. Acquisti, and H. Heinz. Information revelation and privacy in online social networks. In *Proc. 4th ACM Workshop on Privacy in Electronic Society (WPES)*, pages 71–80. ACM, 2005.

[124] The GAIN Collaborative Research Group. New models of collaboration in genome-wide association studies. `http://www.genome.gov/Pages/About/OD/OPG/NewModel-Gain.pdf`, 2007.

[125] S. Guha, K. Tang, and P. Francis. NOYB: Privacy in online social networks. In *Proc. First ACM SIGCOMM Workshop on Social Networks (WOSN)*, pages 49–54. ACM, 2008.

[126] K. Hafner. And if you liked the movie, a Netflix contest may reward you handsomely. New York Times, 2006.

[127] Peter Haggett and Richard J. Chorley. *Network analysis in geography.* Hodder & Stoughton, 1969.

[128] R. Hanneman and M. Riddle. Introduction to social network methods. Chapter 10: Centrality and power. `http://www.faculty.ucr.edu/~hanneman/nettext/C10_Centrality.html`, 2005.

[129] S. Hansell. AOL removes search data on vast group of web users. New York Times, 2006.

[130] S. Hansell. Does cloud computing mean more risks to privacy? `http://bits.blogs.nytimes.com/2009/02/23/does-cloud-computing-mean-more-risks-to-privacy/`, 2009.

[131] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. In *Proc. 34th*

International Conference on Very Large Data Bases (VLDB), pages 102–114. VLDB Endowment, 2008.

[132] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. Technical report, 2007.

[133] B. Hayes. Connecting the dots: Can the tools of graph theory and social-network studies unravel the next big plot? *American Scientist*, 94(5):400–404, 2006.

[134] M. Hellman. A cryptanalytic time-memory tradeoff. *IEEE Transactions on Information Theory*, 26:401–406, 1980.

[135] W. Hwang, T. Kim, M. Ramanathan, and A. Zhang. Bridging centrality: Graph mining from element level to group level. In *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, pages 336–344. ACM, 2008.

[136] IEEE. IEEE P1363: Standard specifications for public-key cryptography. `http://grouper.ieee.org/groups/1363/`, 2008.

[137] IMDb. The Internet Movie Database. `http://www.imdb.com/`, 2007.

[138] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *Proc. Advances in Cryptology - CRYPTO*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.

[139] T. Jagatic, N. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[140] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1), 1906.

[141] I. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin. The design and analysis of graphical passwords. In *Proc. 8th USENIX Security Symposium*, pages 135–150. USENIX, 1999.

[142] A. Juels and J. Brainard. Client puzzles: a cryptographic defense against connection depletion. In *Proc. Network and Distributed System Security Symposium (NDSS)*, pages 151–165. The Internet Society, 1999.

[143] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Proc. Advances in Cryptology – EUROCRYPT*, volume 2045 of *LNCS*, pages 475–494. Springer, 2001.

[144] Testimony of Chris Kelly before the United States Senate Committee on Commerce, Science, and Transportation, "Privacy implications of online advertising" hearing. `http://commerce.senate.gov/public/_files/ChrisKellyFacebookOnlinePrivacyTestimony.pdf`, 2008.

[145] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146. ACM, 2003.

[146] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *Proc. 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 118–127. ACM, 2005.

[147] B. Kerbs. DNA key to decoding human factor. The Washington Post. http://www.washingtonpost.com/wp-dyn/articles/A6098-2005Mar28.html, 2005.

[148] F. Kerschbaum and A. Schaad. Privacy-preserving social network analysis for criminal investigations. In *Proc. 7th ACM Workshop on Privacy in Electronic Society (WPES)*, 2008.

[149] S. Kim, B. Kim, and S. Park. Comments on password-based private key download protocol of NDSS'99. *Electronics Letters*, 35:1937–1938, 1999.

[150] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. In *Proc. 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 86–91. ACM, 2000.

[151] A. Korolova, R. Motwani, S. Nabar, and Y. Xu. Link privacy in social networks. In *Proc. 24th International Conference on Data Engineering (ICDE)*, pages 1355–1357. IEEE Computer Society, 2008.

[152] G. Kossinets, J. Kleinberg, and D. Watts. The structure of information pathways in a social communication network. In *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 435–443. ACM, 2008.

[153] B. Krishnamurthy and C. Willis. Characterizing privacy in online social networks. In *Proc. First ACM SIGCOMM Workshop on Social Networks (WOSN)*. ACM, 2008.

[154] M. Kurucz, A. Benczur, K. Csalogany, and L. Lukacs. Spectral clustering in telephone call graphs. In *Proc. 9th WebKDD and First*

SNA-KDD Workshop on Web Mining and Social Network Analysis (WebKDD/SNA-KDD), pages 82–91. ACM, 2007.

[155] K. Kusuda and T. Matsumoto. Optimization of time-memory trade-off cryptanalysis and its application to DES, FEAL-32 and Skipjack. *IEICE Transactions on Fundamentals*, E79-A(1):35–48, 1996.

[156] K. Kusuda and T. Matsumoto. Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. *IEICE Transactions on Communications/Electronics/Information and Systems*, E82A(1):123–129, 1999.

[157] R. Lambiotte, V. Blondel, C. de Kerchove, E. Huens, C. Prieur, Z. Smoreda, and P. Van Dooren. Geographical dispersal of mobile communication networks. `http://arxiv.org/abs/0802.2178`, 2008.

[158] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. 22nd International Conference on Data Engineering (ICDE)*, page 25. IEEE Computer Society, 2006.

[159] J. Leskovec, L. Adamic, and B. Huberman. The dynamics of viral marketing. In *Proc. 7th ACM Conference on Electronic Commerce*, pages 228–237. ACM, 2006.

[160] K. Lewis, J. Kaufman, M. Gonzales, A. Wimmer, and N. Christakis. Tastes, ties, and time: a new social network dataset using Facebook.com. *Social Networks*, 30(4):330–342, 2008. [Note: six research assistants were paid to download friends-only information from Facebook].

[161] R. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

[162] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. 12th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 556–559. ACM Press, 2003.

[163] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality (to appear)*.

[164] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proc. Advances in Cryptology - CRYPTO*, volume 1880 of *LNCS*, pages 36–54. Springer, 2000.

[165] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 07(1):76–80, 2003.

[166] G. D. Linden, J. A. Jacobi, and E. A. Benson. Collaborative recommendations using item-to-item similarity mappings. U.S. Patent 6266649. `http://www.patentstorm.us/patents/6266649/fulltext.html`, 2008.

[167] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 93–106. ACM, 2008.

[168] M. Lucas and N. Borisov. flyByNight: Mitigating the privacy risks of

social networking. In *Proc. 7th ACM Workshop on Privacy in Electronic Society (WPES)*, pages 1–8. ACM, 2008.

[169] S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proc. Security Protocols Workshop*, volume 1361 of *LNCS*, pages 79–90. Springer, 1997.

[170] B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *Proc. Advances in Cryptology - EUROCRYPT*, volume 3027 of *LNCS*, pages 20–39. Springer, 2004.

[171] C. Maag. A hoax turned fatal draws anger but no charges. New York Times. `http://www.nytimes.com/2007/11/28/us/28hoax.html`, 2007.

[172] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proc. 22nd International Conference on Data Engineering (ICDE)*, page 24. IEEE Computer Society, 2006.

[173] A. Machanavajjhala, D. Martin, D. Kifer, J. Gehrke, and J. Halpern. Worst case background knowledge. In *Proc. 23rd International Conference on Data Engineering (ICDE)*, pages 126–135. IEEE Computer Society, 2007.

[174] B. Malin and L. Sweeney. How (not) to protect genomic data privacy in a distributed network: Using trail re-identification to evaluate and design anonymity protection systems. *J. of Biomedical Informatics*, 37(3):179–192, 2004.

[175] M. McGlohon, L. Akoglu, and C. Faloutsos. Weighted graphs and disconnected components: Patterns and a generator. In *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 524–532. ACM, 2008.

[176] F. McSherry. Privacy integrated queries: Programming privately using LINQ. Dimacs Workshop on Data Privacy, 2008.

[177] F. McSherry. Privacy integrated queries. In *Proc. ACM SIGMOD International Conference on Management of Data, to appear.* ACM, 2009.

[178] F. McSherry and I. Mironov. Differentially private recommender systems. In *Proc. 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), to appear.* ACM, 2009.

[179] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proc. 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103. IEEE Computer Society, 2007.

[180] Medical News Today. WellNet launches online social networking program for health care coordination. `http://www.medicalnewstoday.com/articles/118628.php`, 2008.

[181] B. Mehta and W. Nejdl. Unsupervised strategies for shilling detection and robust collaborative filtering. *User Modeling and User-Adapted Interaction*, 19(1-2):65–97, 2009.

[182] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[183] E. Mills. Facebook suspends app that permitted peephole. `http://news.cnet.com/8301-10784_3-9977762-7.html`, 2008.

[184] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proc. 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 29–42. ACM, 2007.

[185] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective attack models for shilling item-based collaborative filtering systems. In *Advances in Web Mining and Web Usage Analysis, 7th International Workshop on Knowledge Discovery on the Web (WebKDD)*, volume 4198 of *LNCS*, pages 96–118. Springer, 2005.

[186] F. Monrose, M. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. *International Journal of Information Security*, 1(2):69–93, 2002.

[187] C. Moore and M. E. J. Newman. Epidemics and percolation in small-world networks. *Phys. Rev. E*, 61(5):5678–5682, 2000.

[188] R. Morris and K. Thomson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, 1979.

[189] C. Morrison. VentureBeat: Lotame raises $13M for customizable social media ads. `http://tinyurl.com/65pvux`, 2008.

[190] A. A. Nanavati, S. Gurumurthy, G. Das, D. Chakraborty, K. Dasgupta, S. Mukherjea, and A. Joshi. On the structural properties of massive telecom call graphs: findings and implications. In *Proc. 15th ACM*

International Conference on Information and Knowledge Management (CIKM), pages 435–444. ACM, 2006.

[191] A. Narayanan. Lendingclub.com: A de-anonymization walkthrough. `http://33bits.org/2008/11/12/57/`, 2008.

[192] A. Narayanan and V. Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS)*, pages 364–372. ACM, 2005.

[193] A. Narayanan and V. Shmatikov. Obfuscated databases and group privacy. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS)*, pages 364–372. ACM, 2005.

[194] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proc. IEEE Symposium on Security and Privacy*, pages 111–125. IEEE Computer Society, 2008.

[195] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proc. IEEE Symposium on Security and Privacy, to appear*. IEEE Computer Society, 2009.

[196] Netflix. Netflix Prize. `http://www.netflixprize.com/`, 2006.

[197] Netflix. Netflix Prize: FAQ. `http://www.netflixprize.com/faq`, 2006.

[198] H. Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, 79(1):119–158, 2004.

[199] P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In

Proc. Advances in Cryptology - CRYPTO, volume 2729 of *LNCS*, pages 617–630. Springer, 2003.

[200] Office of Compliance and SUNY Downstate Medical Center Audit Services. HIPAA audit physical rounds checklist. `http://www.downstate.edu/hipaa/audit.html`, 2003.

[201] N. O'Neill. Senate begins discussing privacy implications of online advertising. `http://tinyurl.com/5aqqhe`, 2008.

[202] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A.-L. Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, 2007.

[203] Openwall Project. Wordlists collection. `http://www.openwall.com/wordlists/`.

[204] Openwall Project. John the Ripper password cracker. `http://www.openwall.com/john/`, 2005.

[205] Oracle Technical White Paper. The virtual private database in oracle9ir2. `http://www.oracle.com/technology/deploy/security/oracle9ir2/pdf/VPD9ir2twp.pdf`, 2002.

[206] Parliament of Canada. Bill C-6. `http://www2.parl.gc.ca/HousePublications/Publication.aspx?pub=bill&doc=c-6`, 2000. Commonly known as the Personal Information Protection and Electronic Documents Act.

[207] S. Patel. Number theoretic attacks on secure password schemes. In *Proc. IEEE Symposium on Security and Privacy*, page 236. IEEE Computer Society, 1997.

[208] A. Patriquin. Connecting the social graph: Member overlap at OpenSocial and Facebook. `http://tinyurl.com/ynp7t4`, 2007.

[209] B. Pinkas. Cryptographic techniques for privacy preserving data mining. *SIGKDD Explorations*, 4(2):12–19, 2002.

[210] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *Proc. 9th ACM Conference on Computer and Communications Security (CCS)*, pages 161–170. ACM, 2002.

[211] Plaxo. Building an open social graph. `http://www.plaxo.com/info/opensocialgraph`, 2007.

[212] H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proc. 3rd IEEE International Conference on Data Mining (ICDM)*, pages 625–628. IEEE, 2003.

[213] B. Popescu, B. Crispo, and A. Tanenbaum. Safe and private data sharing with Turtle: Friends team-up and beat the system. In *Proc. Cambridge Workshop on Security Protocols*, 2004.

[214] International HapMap project. How are ethical issues being addressed? `http://www.hapmap.org/ethicalconcerns.html.en`.

[215] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[216] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5:54–62, 2001.

[217] D. Recordon. Is SocialMedia overstepping Facebook's privacy line? `http://radar.oreilly.com/2008/07/is-socialmedia-overstepping-fa.html`, 2008.

[218] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 61–70. ACM, 2002.

[219] R. Rivest. Partial-match retrieval algorithms. *SIAM Journal of Computing*, 5(1):19–50, 1976.

[220] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 551–562. ACM, 2004.

[221] T. Rohan, T. Tunguz-Zawislak, S. Sheffer, and J. Harmsen. Network node ad targeting. U.S. Patent Application 0080162260, 2008.

[222] R. Rumford. Facebook applications break 10000. `http://tinyurl.com/5hnqh9`, 2007.

[223] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proc. Advances in Cryptology – EUROCRYPT*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[224] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. 10th International Conference on World Wide Web (WWW)*, pages 285–295. ACM, 2001.

[225] E. Schonfeld. TechCrunch: Twitter starts blacklisting spammers. `http://www.techcrunch.com/2008/05/07/twitter-starts-blacklisting-spammers/`, 2008.

[226] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proc. 2nd International Workshop on Privacy Enhancing Technologies (PET)*, volume 2482 of *LNCS*, pages 41–53. Springer, 2003.

[227] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[228] Z. Shuanglei. Project RainbowCrack. `http://www.antsight.com/zsl/rainbowcrack/`, 2005.

[229] G. Simmel. *Soziologie*. Duncker & Humblot, 1908. [Note: Simmel proposed a new and quantitative approach to sociology, one that would fall under Social Network Analysis in modern terms.].

[230] R. Singel. Private Facebook pages are not so private. `http://www.wired.com/software/webservices/news/2007/06/facebookprivacysearch`, 2007.

[231] C. Soghoian. Widespread cell phone location snooping by NSA? `http://news.cnet.com/8301-13739_3-10030134-46.html`, 2008.

[232] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proc. IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.

[233] F. Standaert, G. Rouvroy, J.J. Quisquater, and J. Legat. A time/memory tradeoff using distinguished points: New analysis and FPGA results. In *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, volume 2523 of *LNCS*, pages 593–609. Springer, 2002.

[234] Z. Stone, T. Zickler, and T. Darrell. Autotagging Facebook: Social network context improves photo annotation. In *Workshop on Internet Vision*. IEEE, 2008.

[235] L. Story. A company promises the deepest data mining yet. New York Times. `http://www.nytimes.com/2008/03/20/business/media/20adcoside.html`, 2008.

[236] S. Stubblebine and P. van Oorschot. Addressing online dictionary attacks with login histories and humans-in-the-loop. In *Proc. Financial Cryptography, 4th International Conference*, volume 3110 of *LNCS*, pages 39–53. Springer, 2004.

[237] X. Suo, Y. Zhu, and G. S. Owen. Graphical passwords: A survey. In *Proc. 21st Computer Security Applications Conference (ACSAC)*, pages 463–472. IEEE Computer Society, 2005.

[238] G. Swamynathan, C. Wilson, B. Boe, B. Zhao, and K. Almeroth. Can social networks improve e-commerce: A study on social marketplaces. In

Proc. First ACM SIGCOMM Workshop on Social Networks (WOSN), pages 1–6. ACM, 2008.

[239] L. Sweeney. Weaving technology and policy together to maintain confidentiality. *J. of Law, Medicine and Ethics*, 25(2–3):98–110, 1997.

[240] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International J. of Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[241] L. Sweeney. k-anonymity: A model for protecting privacy. *International J. of Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[242] M. Sweney. Google and Viacom reach deal over YouTube user data. Guardian. `http://tinyurl.com/59b3ou`, 2008.

[243] Techcrunch. Facebook news feed reports on you behind your back. `http://www.techcrunch.com/2008/04/14/facebook-newsfeed-reports-on-you-behind-your-back/`, 2007.

[244] Techdirt. Is a fake Facebook profile illegal? `http://techdirt.com/articles/20080604/0152031306.shtml`, 2008.

[245] J. Thornton. Collaborative filtering research papers. `http://jamesthornton.com/cf/`, 2006.

[246] J. Thorpe and P. van Oorschot. Graphical dictionary and the memorable space of graphical passwords. In *Proc. 13th USENIX Security Symposium*, pages 135–150. USENIX, 2004.

[247] R. Topolski. NebuAd and partner ISPs: Wiretapping, forgery and browser hijacking. http://www.freepress.net/files/NebuAd_Report.pdf, 2008.

[248] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[249] United States Code. The Privacy Act of 1974 and amendments. http://epic.org/privacy/laws/privacy_act.html, 2005.

[250] United States Department of Health and Human Services. Standards for privacy of individually identifiable health information; Final rule. http://www.hhs.gov/ocr/hipaa/privrulepd.pdf, 2002. Commonly known as the HIPAA Privacy Rule.

[251] United States Justice Department. The Privacy Act of 1974. http://www.usdoj.gov/oip/privstat.htm, 1974.

[252] United States Senate. Text of the Consumer Privacy Protection Act of 2005. http://www.govtrack.us/congress/billtext.xpd?bill=h109-1263, 2005.

[253] United States Senate. Text of the Online Privacy Protection Act of 2005. http://www.govtrack.us/congress/billtext.xpd?bill=h109-84, 2005.

[254] United States Senate. Text of the Privacy Act of 2005. http://www.govtrack.us/congress/billtext.xpd?bill=s109-116, 2005.

[255] X. Wang and M. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proc. IEEE Symposium on Security and Privacy*, pages 78–92. IEEE Computer Society, 2003.

[256] H. Wee. On obfuscating point functions. In *Proc. 37th ACM Symposium on Theory of Computing (STOC)*, pages 523–532. ACM, 2005.

[257] E. Weinstein. Phase transition. `http://mathworld.wolfram.com/PhaseTransition.html`.

[258] Wikipedia. Phorm — Wikipedia, the free encyclopedia, 2008. [Online; accessed April-2008].

[259] L. Willenborg and T. de Waal. *Statistical Disclosure Control in Practice*. Springer, New York, 1996.

[260] G. Wills. NicheWorks — Interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics*, 8(2):190–212, 1999.

[261] J. Winter. Disgraced former NBA referee Tim Donaghy's phone calls to second ref raise questions. `http://www.foxnews.com/story/0,2933,381842,00.html`, 2008.

[262] PC World. Facebook's beacon more intrusive than previously thought. `http://www.pcworld.com/article/id,140182-c,onlineprivacy/article.html`, 2007.

[263] T. Wu. A real-world analysis of Kerberos password security. In *Proc. Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 1999.

[264] A. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE, 1982.

[265] A. Yao. How to generate and exchange secrets. In *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167. IEEE, 1986.

[266] H. Yu. Freedom to Tinker: Bad Phorm on privacy. `http://tinyurl.com/6qkstm`, 2008.

[267] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against sybil attacks. In *Proc. IEEE Symposium on Security and Privacy*, pages 3–17. IEEE Computer Society, 2008.

[268] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *Proc. First SIGKDD International Workshop on Privacy, Security, and Trust in KDD (PinKDD)*. ACM, 2007.

[269] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *Proc. 24th International Conference on Data Engineering (ICDE)*, pages 506–515. IEEE Computer Society, 2008.

[270] H. Zhu, T. Liu, J. Liu, and G. Chang. A method for making three-party password-based key exchange resilient to server compromise. In *Proc. 3rd International Conference on International Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, pages 546–549. IEEE Computer Society, 2007.

Vita

Arvind Narayanan was born in Madras, India in 1981. He received a Bachelor of Technology and Master of Technology degrees in Computer Science from the Indian Institute of Technology, Madras, India. He enrolled in the Ph.D program at the University of Texas at Austin in 2004.

Permanent Address: 3401 Red River St. #232 Austin TX 78705 USA

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.